Attorney Docket No. 1341.1192

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Hidenobu ITO et al.

Application No.: Group Art Unit: Unassigned

Filed: February 26, 2004 Examiner: Unassigned

For: APPARATUS FOR AND METHOD OF CREATING COMPUTER PROGRAM SPECIFICATIONS, AND COMPUTER PROGRAM PRODUCT

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). 2003-355695

Filed: October 15, 2003

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: Feb 25, 2004

By: _____
Mark J. Henry
Registration No. 36,162

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

PFJA-030[illegible]-US

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年１０月１５日

出願番号
Application Number: 特願２００３－３５５６９５

[ST. 10/C]: [ＪＰ２００３－３５５６９５]

出願人
Applicant(s): 富士通株式会社

２００３年１２月１８日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３１０５３１４

【書類名】　特許願
【整理番号】　0352339
【提出日】　平成15年10月15日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/06　530
G06F　9/06　540
G06F 19/00
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　伊藤　栄信
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　松尾　昭彦
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　五味　俊明
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　上村　学
【特許出願人】
【識別番号】　000005223
【氏名又は名称】　富士通株式会社
【代理人】
【識別番号】　100089118
【弁理士】
【氏名又は名称】　酒井　宏明
【手数料の表示】
【予納台帳番号】　036711
【納付金額】　21,000円
【提出物件の目録】
【物件名】　特許請求の範囲　1
【物件名】　明細書　1
【物件名】　図面　1
【物件名】　要約書　1
【包括委任状番号】　9717671

【書類名】特許請求の範囲
【請求項１】
ソースプログラムを解析して仕様書を生成する仕様書生成プログラムであって、
前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、
前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、
前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と
をコンピュータに実行させることを特徴とする仕様書生成プログラム。
【請求項２】
前記処理概要情報生成手順は、特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを入出力付構造情報から抽出して前記処理概要情報を生成し、
前記文書生成手順は、前記特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを用いてプログラム仕様書を生成することを特徴とする請求項１に記載の仕様書生成プログラム。
【請求項３】
前記文書生成手順は、サブルーチンの呼出階層を列とし、各サブルーチンの名前をサブルーチンの呼出階層値に対応する列に表示する表形式の呼出構造図をプログラム仕様書に含めることを特徴とする請求項２に記載の仕様書生成プログラム。
【請求項４】
前記入出力付構造情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、
前記処理概要情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、
前記文書生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成することを特徴とする請求項１に記載の仕様書生成プログラム。
【請求項５】
前記文書生成手順は、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出するコメント抽出手順と、
新たに生成するプログラム仕様書の所定の位置に、前記コメント抽出手順により抽出されたコメントを継承するコメント継承手順とを
さらにコンピュータに実行させることを特徴とする請求項１、２または４に記載の仕様書生成プログラム。
【請求項６】
バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出するステップ入出力情報抽出手順と、
前記ステップ入出力情報抽出手順により抽出されたジョブステップの入出力情報に基づいて前記バッチジョブ全体の入力情報および出力情報を特定するジョブ入出力情報特定手順と、
前記ジョブ入出力情報特定手順により特定された入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出するプログラム情報抽出手順と、
前記ジョブ入出力情報特定手順により特定された入力情報および出力情報、ならびに前記プログラム情報抽出手順により抽出されたプログラムの情報を用いて前記バッチジョブのバッチジョブ処理概要書を生成するバッチジョブ処理概要書生成手順と
をさらにコンピュータに実行させることを特徴とする請求項１、２または４に記載の仕様書生成プログラム。
【請求項７】

画面についての情報を定義した画面定義体を解析して画面遷移情報を作成する画面遷移情報作成手順と、
前記画面遷移情報作成手順により作成された画面遷移情報を用いて画面遷移図を生成する画面遷移図生成手順と
をさらにコンピュータに実行させることを特徴とする請求項１、２または４に記載の仕様書生成プログラム。
【請求項８】
ソースプログラムを解析して仕様書を生成する仕様書生成プログラムを記録したコンピュータ読み取り可能な記録媒体であって、
前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、
前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、
前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と
をコンピュータに実行させる仕様書生成プログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。
【請求項９】
ソースプログラムを解析して仕様書を生成する仕様書生成装置であって、
前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手段と、
前記入出力付構造情報生成手段により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手段と、
前記処理概要情報生成手段により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手段と
を備えたことを特徴とする仕様書生成装置。
【請求項１０】
ソースプログラムを解析して仕様書を生成する仕様書生成方法であって、
前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成工程と、
前記入出力付構造情報生成工程により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成工程と、
前記処理概要情報生成工程により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成工程と
を含んだことを特徴とする仕様書生成方法。

【書類名】明細書
【発明の名称】仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法
【技術分野】
【０００１】
この発明は、ソースプログラムを解析して仕様書を生成する仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法に関し、特に、ソフトウェアの処理の全体像の把握を可能とし、もってソフトウェアの保守を容易にすることができる仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法に関するものである。
【背景技術】
【０００２】
ソフトウェアの保守や、そのソフトウェアを元にした新たなソフトウェアの開発を行う際には、プログラムの処理内容を理解する必要がある。しかし、計算機の処理に適する形に設計されたプログラム言語は人間が直接読んでも内容を理解するのが難しく、特に多数のプログラムから構成される大規模なソフトウェアにおいては、ソースプログラムだけから全体を把握するのはほとんど不可能である。このため、ソフトウェア保守や開発の際には、プログラムの処理内容を人間が理解しやすい形で記述した仕様書を用いるのが一般的となっている。
【０００３】
特に、運用中のソフトウェアの保守を請け負うＡＰＭ(Application Portfolio Management)サービスでは、開発・運用経験の無いソフトウェアを対象として保守を行うため、仕様書の存在が不可欠となる。
【０００４】
しかし、ソフトウェアの仕様書は、ほとんどの場合、プログラムの開発の際に作成されるものの、開発後に加えられた修正や仕様変更が反映されない場合も多く、内容が必ずしも信頼できるとは限らない。また、仕様書そのものが残っていない場合もあり、信頼できる仕様書を整備することが、保守作業を行う上で非常に重要となる。
【０００５】
また、仕様情報をプログラムのコメント文として記入しておくことがよく行われるが、プログラムの修正にあわせてコメントの内容も正しく修正されているという保証がないため、なんら解決とはなっていない。
【０００６】
そこで、プログラムの呼出構造などを機械的に解析して仕様書を自動生成することが従来から試みられている（例えば、特許文献１、非特許文献１参照。）。このような仕様書の自動生成では、生成される仕様書の内容がプログラムの内容と一致していることを保証することができ、信頼できる仕様書を得ることができる。
【０００７】
【特許文献１】特開２００２－２１５３９１号公報
【非特許文献１】「ＳＩＭＰＬＩＡシリーズ」、［平成１５年９月２５日検索］、インターネット＜URL: http://software.fujitsu.com/jp/simplia/＞
【発明の開示】
【発明が解決しようとする課題】
【０００８】
しかしながら、従来のように、プログラムの呼出構造などを解析して表示するだけでは、プログラムの詳細な構造はわかっても、プログラムの理解に最も必要な処理概要を把握することができないという問題があった。特に、プログラムの概要理解に最も重要な、プログラムの呼出構造と入出力情報を関連づける情報がないという問題があった。
【０００９】
また、一度生成した仕様書に対して利用者がプログラムの理解の助けとして書き込みを

行った場合、プログラム修正に伴って仕様書を再度自動生成すると、新しく生成された仕様書には書き込んだ情報が欠落してしまうため、人間が内容を書き写す必要があるという問題があった。

【００１０】

また、複数のプログラムの実行によって処理が行われるジョブ全体の処理を理解する場合には、個々のプログラムの仕様書ではなく、ジョブ全体の処理概要について記載した仕様書が必要となるが、このような仕様書は、個々のプログラムを解析しただけでは生成することができないという問題があった。

【００１１】

この発明は、上述した従来技術による問題点を解消するためになされたものであり、ソフトウェアの処理の全体像の把握を可能とし、もってソフトウェアの保守を容易にすることができる仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法を提供することを目的とする。

【課題を解決するための手段】

【００１２】

上述した課題を解決し、目的を達成するため、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成プログラムであって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順とをコンピュータに実行させることを特徴とする。

【００１３】

また、本発明は、上記の発明において、前記入出力付構造情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、前記処理概要情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、前記文書生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成することを特徴とする。

【００１４】

また、本発明は、上記発明において、前記文書生成手順は、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出するコメント抽出手順と、新たに生成するプログラム仕様書の所定の位置に、前記コメント抽出手順により抽出されたコメントを継承するコメント継承手順とをさらにコンピュータに実行させることを特徴とする。

【００１５】

また、本発明は、上記発明において、前記ソースプログラムを構成する文をまとめることにより該ソースプログラムのプログラム概要情報を生成するプログラム概要情報生成手順と、前記プログラム概要情報生成手順により生成されたプログラム概要情報から自然言語によるプログラム概要文を生成するプログラム概要文生成手順と、前記プログラム概要文生成手順により生成されたプログラム概要文を用いて前記ソースプログラムのプログラム概要書を生成するプログラム概要書生成手順とをさらにコンピュータに実行させることを特徴とする。

【００１６】

また、本発明は、上記発明において、バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出するステップ入出力情報抽出手順と、前記ステップ入出力情報抽出手順により抽出されたジョブステップの入出力情報に基づいて前記バッチジョブ全体の入力情報および出力情報を特定するジョブ入出力情報特定手順と、前記ジョブ入出力情報特定手順により特定された入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び

出されるプログラムの情報を抽出するプログラム情報抽出手順と、前記ジョブ入出力情報特定手順により特定された入力情報および出力情報、ならびに前記プログラム情報抽出手順により抽出されたプログラムの情報を用いて前記バッチジョブのバッチジョブ処理概要書を生成するバッチジョブ処理概要書生成手順とをさらにコンピュータに実行させることを特徴とする。

【００１７】

また、本発明は、上記発明において、画面についての情報を定義した画面定義体を解析して画面遷移情報を作成する画面遷移情報作成手順と、前記画面遷移情報作成手順により作成された画面遷移情報を用いて画面遷移図を生成する画面遷移図生成手順とをさらにコンピュータに実行させることを特徴とする。

【００１８】

また、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成プログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順とをコンピュータに実行させる仕様書生成プログラムを記録したことを特徴とする。

【００１９】

また、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成装置であって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手段と、前記入出力付構造情報生成手段により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手段と、前記処理概要情報生成手段により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手段とを備えたことを特徴とする。

【００２０】

また、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成方法であって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成工程と、前記入出力付構造情報生成工程により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成工程と、前記処理概要情報生成工程により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成工程とを含んだことを特徴とする。

【発明の効果】

【００２１】

本発明によれば、ソースプログラムを解析してソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成し、生成した入出力付構造情報から一部の情報を抽出してソースプログラムの処理概要情報を生成し、生成した処理概要情報を用いてソースプログラムのプログラム仕様書を生成するよう構成したので、ソフトウェアの処理の全体像の把握を可能とし、もってソフトウェアの保守を容易にすることができるという効果を奏する。

【００２２】

また、本発明によれば、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成するよう構成したので、どのような条件の際に入出力が行われているかの把握を容易にすることができるという効果を奏する。

【００２３】

また、本発明によれば、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出し、新たに生成するプログラム仕様書の所定の位置に、抽出したコメントを継承するよう構成したので、利用者によりプログラム仕様書に追記されたコメントを自動的に継承することができるという効果を奏する。

【００２４】

また、本発明によれば、ソースプログラムを構成する文をまとめることによりソースプログラムのプログラム概要情報を生成し、生成したプログラム概要情報から自然言語によるプログラム概要文を生成し、生成したプログラム概要文を用いてソースプログラムのプログラム概要書を生成するよう構成したので、プログラムの処理内容の理解を容易し、理解に要する時間を短縮することができるという効果を奏する。

【００２５】

また、本発明によれば、バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出し、抽出したジョブステップの入出力情報に基づいてバッチジョブ全体の入力情報および出力情報を特定し、特定した入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出し、特定された入力情報および出力情報、ならびに抽出したプログラムの情報を用いてバッチジョブのバッチジョブ処理概要書を生成するよう構成したので、バッチジョブ処理全体の概要の把握を容易にすることができるという効果を奏する。

【００２６】

また、本発明によれば、画面についての情報を定義した画面定義体を解析して画面遷移情報を作成し、作成した画面遷移情報を用いて画面遷移図を生成するよう構成したので、オンラインシステムの処理概要の把握を容易にすることができるという効果を奏する。

【発明を実施するための最良の形態】

【００２７】

以下に添付図面を参照して、この発明に係る仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法の好適な実施例を詳細に説明する。なお、本実施例では、本発明をＣＯＢＯＬで開発されたプログラムに適用した場合を中心に説明する。

【実施例１】

【００２８】

まず、本実施例１に係る仕様書生成装置の構成について説明する。図１は、本実施例１に係る仕様書生成装置の構成を示す機能ブロック図である。同図に示すように、この仕様書生成装置１００は、プログラムソースからプログラム仕様書を作成する処理を行う制御部１１０と、制御部１１０による処理の途中結果などを記憶する記憶部１２０とを有する。

【００２９】

制御部１１０は、構文解析部１１１と、構造抽出部１１２と、条件分岐情報抽出部１１３と、入出力データ抽出部１１４と、構造－入出力データ関連生成部１１５と、処理概要生成部１１６と、ドキュメント生成部１１７とを有し、記憶部１２０は、構文解析情報記憶部１２１と、サブルーチン情報記憶部１２２と、入出力情報記憶部１２３と、処理概要情報記憶部１２４とを有する。

【００３０】

構文解析部１１１は、プログラムソースを読み込み、読み込んだプログラムソース中の文や変数の情報を抽出して構文解析情報を生成する処理部であり、構文解析情報記憶部１２１は、構文解析部１１１が生成した構文解析情報を記憶する記憶部である。

【００３１】

構造抽出部１１２は、構文解析情報記憶部１２１から構文解析情報を読み出し、プログラム内のサブルーチン呼出における呼出先の解決を行い、実行順序にしたがったプログラムの呼出構造をサブルーチン情報として抽出する処理部である。

【００３２】
図２は、構造抽出部１１２が抽出するサブルーチン呼出の構造を説明するための説明図である。同図は、あるプログラムが「サブルーチン１」を呼び出し、「サブルーチン１」が「サブルーチン２」を呼び出し、「サブルーチン２」が「サブルーチン３」を呼び出す場合を示している。
【００３３】
なお、ここでは、サブルーチンはプログラムが他のプログラムを呼び出す場合（ＣＯＢＯＬではＣＡＬＬ）と、プログラムの内部に含まれる処理の塊を呼び出す場合（ＣＯＢＯＬではＰＥＲＦＯＲＭ）の両方を示している。
【００３４】
そして、「サブルーチン３」の実行が終了すると、「サブルーチン２」に制御が戻り、「サブルーチン２」の実行が終了すると「サブルーチン１」に制御が戻り、「サブルーチン１」の実行が終了すると、もとのプログラムに制御が戻る。構造抽出部１１２は、このようなプログラムの構造を抽出し、サブルーチン情報を生成する。
【００３５】
サブルーチン情報記憶部１２２は、構造抽出部１１２が生成したサブルーチン情報を記憶する記憶部である。図３は、サブルーチン情報記憶部１２２が記憶するサブルーチン情報のデータ構造の一例を示す図である。
【００３６】
同図に示すように、このサブルーチン情報記憶部１２２が記憶するサブルーチン情報には、サブルーチンの名前を示すサブルーチン名と、呼出の深さを示す呼出階層値と、サブルーチンが呼び出されるときの条件のリストである呼出条件リストと、サブルーチンが行う入出力の情報のリストである入出力情報リストと、このサブルーチンが呼び出すサブルーチンのサブルーチン情報のリストである呼出リストが含まれる。ここで、サブルーチン名は、メインルーチンなどサブルーチン名がない場合には、”” （空文字）がセットされる。
【００３７】
また、呼出リストは、同様のデータ構造を有するサブルーチン情報のリストである。図４は、サブルーチン情報のリストのつながり（ツリー構造）を説明するための説明図である。同図は、呼出階層値が「１」であるサブルーチン情報の呼出リストは、呼出階層が「２」であるサブルーチンのサブルーチン情報のリストで構成され、呼出階層値が「２」であるサブルーチン情報の呼出リストは、呼出階層が「３」であるサブルーチンのサブルーチン情報のリストで構成され、全体としてサブルーチン情報がツリー構造を構成することを示している。
【００３８】
図５は、図３に示した呼出条件リストのデータ構造の一例を示す図である。同図に示すように、呼出条件リストを構成する各呼出条件には、ＩＦ文やＳＷＩＴＣＨ文における分岐条件、ループ文におけるループ条件などの条件の種別を示す条件種別と、条件を記述した条件記述とが含まれる。例えば、条件記述としては、「ｉ＜１００」などがある。
【００３９】
図６は、図３に示した入出力情報リストのデータ構造の一例を示す図である。同図に示すように、入出力情報リストを構成する各入出力情報には、入力か出力かを示す入出力区別と、入出力の対象がデータベースかファイルかなどを示す対象種類と、入出力の対象の名前を示す対象名と、入出力に使用する入出力命令を示す使用命令文と、対象に固有なその他の情報を示すオプション情報とが含まれる。
【００４０】
条件分岐情報抽出部１１３は、構文解析情報記憶部１２１に記憶された構文解析情報から図５に示した呼出条件リストを生成する処理部である。この条件分岐情報抽出部１１３によって生成された呼出条件リストは、サブルーチン情報記憶部１２２に格納される。
【００４１】

入出力データ抽出部１１４は、構文解析情報記憶部１２１に記憶された構文解析情報を読み出し、各サブルーチンに対して図６に示した入出力情報リストを生成する処理部であり、入出力情報記憶部１２３は、入出力データ抽出部１１４により生成された入出力情報リストをサブルーチン名と対応させて記憶する記憶部である。

【００４２】

構造－入出力データ関連生成部１１５は、入出力情報記憶部１２３に記憶された入出力情報リストとプログラムの呼出構造とを対応させる処理部である。具体的には、この構造－入出力データ関連生成部１１５は、入出力情報記憶部１２３に記憶された入出力情報リストをサブルーチン情報記憶部１２２の中の対応するサブルーチン情報の入出力情報リストとして格納する。

【００４３】

この構造－入出力データ関連生成部１１５が、入出力情報リストとプログラムの呼出構造とを対応させることによって、仕様書生成装置１００は、プログラムの呼出構造と入出力情報を関連付けたプログラム仕様書を生成することができる。

【００４４】

処理概要生成部１１６は、構造－入出力データ関連生成部１１５により入出力情報リストが格納されたサブルーチン情報からプログラムの処理概要情報を生成する処理部である。処理概要情報記憶部１２４は、処理概要生成部１１６により生成された処理概要情報を記憶する記憶部である。

【００４５】

処理概要生成部１１６は、

（１）呼出階層値が特定の範囲（例えば、２および３）にあるサブルーチン情報だけを抽出する

（２）特定の単語(例えば、「エラー」、「障害」、「初期化」など）をサブルーチン名に含むサブルーチン情報以下のサブルーチン情報は除く

（３）入出力を行うサブルーチンだけを残す

（４）ループ処理を含むサブルーチン情報を残す

などの処理によって処理概要情報を生成する。

【００４６】

図７は、処理概要生成部１１６が生成する処理概要情報を説明するための説明図である。同図は、呼出階層値が「２および３」のサブルーチン情報だけを抽出して処理概要情報を作成する場合を示している。

【００４７】

このように、この処理概要生成部１１６が、サブルーチン情報からプログラムの処理概要情報を生成することによって、仕様書生成装置１００は、プログラムの呼出構造と入出力情報が関連づけられた処理概要を示すプログラム仕様書を生成することかできる。

【００４８】

また、この処理概要生成部１１６は、プログラムの入出力ファイルを図示するために必要な情報、プログラムが使用する共通領域についての情報、プログラムが入出力するファイルについての情報を生成し、処理概要情報記憶部１２４に格納する。

【００４９】

ドキュメント生成部１１７は、処理概要情報記憶部１２４に記憶された処理概要情報を用いてプログラム仕様書を生成する処理部である。具体的には、このドキュメント生成部１１７は、処理概要情報を所定の形式に整形し、プログラム仕様書として出力する。

【００５０】

図８は、ドキュメント生成部１１７が生成するプログラム仕様書の一例を示す図である。同図に示すように、このプログラム仕様書は、プログラム名と、プログラムを格納するファイル名と、［コメント欄］と、［入出力関連図］と、［共通領域］と、［ファイル情報］と、［入出力とセクション名］と、［処理構造図］とを有する。

【００５１】

［コメント欄］は、利用者が業務情報、運用ノウハウなどのコメントを追記する部分であり、［入出力関連図］は、プログラムの入出力ファイルを図示する部分である。［共通領域］は、プログラムが使用する共通領域のレコード名を示す部分であり、［ファイル情報］は、プログラムが入出力するファイルについての情報を示す部分である。

【００５２】

［入出力とセクション名］は、処理概要情報として抽出されたサブルーチン名および抽出されたサブルーチン情報にふくまれる入出力情報を示す部分である。ここで、セクションとは、サブルーチンの一種であるが、ここでは、セクションでサブルーチンを代用している。この例では、呼出階層値が「２および３」のセクションについての情報が処理概要情報として示されている。

【００５３】

すなわち、呼出階層値が「２」であるセクションとして「パラメタチェック処理」、「読込処理」および「詳細処理」が示されており、「詳細処理」が呼び出す呼出階層値が「３」であるセクションとして、カッコ内に「データチェック処理」、「調停額確定Ｆ読込処理」、「名義Ｆ読込処理」および「編集処理」が示されている。

【００５４】

また、これらのセクションのサブルーチン情報に含まれる入出力情報には、入力として「ＴＪＳＣ．申込ファイル」、「ＴＪＳＣ．入居ファイル」、「ＴＪＳＣ．住戸マスタ」、「ＴＪＳＣ．使用料区分マスタ」、「ＴＪＳＣ．調停額確定ファイル」、「ＴＪＳＣ．名義ファイル」、「ＴＪＳＣ．コードマスタ」、「ＳＣ＿Ｂ１０１００１．ＭＢ１０１０１２」、「ＳＣ＿Ｂ１０１００１．ＭＢ１０１０１４」および「ＩＮ０１」があり、出力として「ＴＪＣＣＦ０５１」がある。

【００５５】

［処理構造図］は、処理概要情報として抽出されたサブルーチンの呼出構造を示す部分である。この例では、呼出階層値が「２」であるセクションとして「パラメタチェック処理」、「読込処理」および「詳細処理」が示されており、「詳細処理」が呼び出す呼出階層値が「３」の「データチェック処理」が示されている。

【００５６】

次に、構造抽出部１１２によるサブルーチン情報生成処理および条件分岐情報抽出部１１３による条件分岐情報抽出処理の処理手順について説明する。図９は、構造抽出部１１２によるサブルーチン情報生成処理および条件分岐情報抽出部１１３による条件分岐情報抽出処理の処理手順を示すフローチャートである。なお、同図において、破線で囲まれた処理が条件分岐情報抽出部１１３による条件分岐情報抽出処理であり、他の部分は、構造抽出部１１２によるサブルーチン情報生成処理である。

【００５７】

同図に示すように、まず、構造抽出部１１２が構文解析情報記憶部１２１に記憶された構文解析情報からサブルーチンごとのプログラム位置を取得し（ステップＳ１１１）、また、プログラム全体のスタート位置を検出する（ステップＳ１１２）。

【００５８】

そして、最初のサブルーチン情報を作成し（ステップＳ１１３）、スタート位置の次のプログラム文の構造を調べ（ステップＳ１１４）、サブルーチン呼出であるか否かを判定する（ステップＳ１１５）。ここで、サブルーチン呼出を行うプログラム文としては、ＰＥＲＦＯＲＭ文やＣＡＬＬ文がある。

【００５９】

そして、サブルーチン呼出である場合には、今のプログラム位置を保持し（ステップＳ１１ｃ）、子サブルーチンに対するサブルーチン情報を作成して呼出リストに登録し（ステップＳ１１ｄ）、呼出条件作業リストがあれば子サブルーチン情報に追加する（ステップＳ１１ｅ）。そして、調べるプログラムの位置をサブルーチンのトップに移動し（ステップＳ１１ｆ）、ステップＳ１１４に戻って次のプログラム文の構造を調べる。

【００６０】

一方、サブルーチン呼出でない場合には、条件分岐情報抽出部１１３が条件文であるか否かを判定し（ステップＳ１１６）、条件文である場合には、呼出条件作業リストに条件を追加し（ステップＳ１１７）、ステップＳ１１４に戻って次のプログラム文の構造を調べる。

【００６１】

また、条件文でない場合には、条件文を抜けたか否かを判定し（ステップＳ１１８）、条件文を抜けた場合には、呼出条件作業リストから条件を一つ削除し（ステップＳ１１９）、ステップＳ１１４に戻って次のプログラム文の構造を調べる。

【００６２】

また、条件文を抜けていない場合には、構造抽出部１１２が、サブルーチンの終わりか否かを判定し（ステップＳ１１ａ）、サブルーチンの終わりでない場合には、プログラムの終わりか否かを判定し（ステップＳ１１ｂ）、プログラムの終わりでない場合には、ステップＳ１１４に戻って次のプログラム文の構造を調べ、プログラムの終わりの場合には、処理を終了する。

【００６３】

一方、サブルーチンの終わりの場合には、操作するサブルーチン情報を親サブルーチンに戻し（ステップＳ１１ｇ）、保持していた親のプログラム位置に、調べるプログラムの位置を戻し（ステップＳ１１ｈ）、ステップＳ１１４に戻って次のプログラム文の構造を調べる。

【００６４】

このように、構造抽出部１１２が、プログラムの中のサブルーチン呼出およびサブルーチンの終わりを調べてサブルーチン情報を作成することによって、プログラムからプログラムの呼出構造を抽出することができる。

【００６５】

また、条件分岐情報抽出部１１３が条件文を調べて呼出条件作業リストに保持し、サブルーチン呼出があった場合に、構造抽出部１１２が、呼出条件作業リストの条件をサブルーチン情報の呼出条件リストに登録することによって、サブルーチン呼出の呼出条件を抽出することができる。

【００６６】

次に、入出力データ抽出部１１４による入出力情報抽出処理の処理手順について説明する。図１０は、入出力データ抽出部１１４による入出力情報抽出処理の処理手順を示すフローチャートである。

【００６７】

同図に示すように、この入出力データ抽出部１１４は、構文解析情報記憶部１２１に記憶された構文解析情報からプログラム全体のスタート位置を検出し（ステップＳ１２１）、また、現在のサブルーチン名を保持する（ステップＳ１２２）。

【００６８】

そして、次のプログラム文を調べ（ステップＳ１２３）、入出力文か否かを判定する（ステップＳ１２４）。ここで、入出力文としては、ＲＥＡＤ文、ＷＲＩＴＥ文などがある。

【００６９】

そして、入出力文である場合には、今のサブルーチンで入出力情報記憶部１２３にサブルーチン情報を作成済みか否かを調べ、サブルーチン情報を作成していない場合には、サブルーチン情報を作成して入出力情報記憶部１２３に登録する（ステップＳ１２７）。

【００７０】

そして、入出力文から入出力情報を作成し（ステップＳ１２８）、入出力情報記憶部１２３に記憶されたサブルーチン情報に入出力情報を追加し（ステップＳ１２９）、ステップＳ１２３に戻って次のプログラム文を調べる。

【００７１】

一方、プログラム文が入出力文でない場合には、サブルーチンが変わったか否かを調べ

（ステップＳ１２５）、サブルーチンが変わった場合には、ステップＳ１２２に戻って、変わったサブルーチンの名前を現在のサブルーチン名として保持する。

【００７２】

また、サブルーチンが変わっていない場合には、プログラムの終わりか否かを調べ（ステップＳ１２６）、プログラムの終わりでなければステップＳ１２３に戻って次のプログラム文を調べ、プログラムの終わりであれば処理を終了する。

【００７３】

このように、この入出力データ抽出部１１４が、プログラムから入出力情報を抽出し、抽出した入出力情報を入出力を行うサブルーチンに対応させて入出力情報記憶部１２３に記憶することによって、サブルーチンと入出力情報を関連づけることができる。

【００７４】

次に、構造－入出力データ関連生成部１１５による入出力情報のサブルーチン情報記憶部１２２への追加処理の処理手順について説明する。図１１は、構造－入出力データ関連生成部１１５による入出力情報のサブルーチン情報記憶部１２２への追加処理の処理手順を示すフローチャートである。

【００７５】

同図に示すように、この構造－入出力データ関連生成部１１５は、サブルーチン情報記憶部１２２に記憶されたサブルーチン情報のツリー構造から一つのサブルーチン情報を取り出す（ステップＳ１３１）。そして、サブルーチン名を取得し（ステップＳ１３２）、入出力情報記憶部１２３に記憶されたサブルーチン情報から同じ名前のサブルーチンを探す（ステップＳ１３３）。

【００７６】

そして、同じ名前のサブルーチンが見つかったか否かを判定し（ステップＳ１３４）、同じ名前のサブルーチンが見つかった場合には、入出力情報記憶部１２３に記憶された入出力情報をサブルーチン情報記憶部１２２の対応するサブルーチン情報に追加する（ステップＳ１３５）。

【００７７】

そして、サブルーチン情報記憶部１２２のサブルーチン情報をすべて取り出したか否かを判定し（ステップＳ１３６）、サブルーチン情報をすべて取り出していない場合には、ステップＳ１３１に戻って次のサブルーチン情報を取り出し、サブルーチン情報をすべて取り出した場合には、処理を終了する。

【００７８】

このように、この構造－入出力データ関連生成部１１５が入出力情報記憶部１２３に記憶された入出力情報をサブルーチン情報記憶部１２２の対応するサブルーチン情報に追加することによって、プログラム呼出構造と入出力情報を関連づけることができる。

【００７９】

なお、サブルーチンとして、プログラムの内部に含まれる処理の塊を呼び出す場合と、別のプログラムを呼び出す場合の両方が混在するとき、階層が非常に深くなる場合がある。このようなとき、別のプログラムを呼び出したその次以降の階層は省略し、入出力情報だけ、その呼び出されるプログラムのサブルーチン情報の入出力情報リストに加えるというような省略方法もある。

【００８０】

次に、処理概要生成部１１６による処理概要生成処理の処理手順について説明する。図１２は、処理概要生成部１１６による処理概要生成処理の処理手順を示すフローチャートである。なお、ここでは、処理概要生成部１１６による処理概要生成処理のうち、呼出階層の値について特定の値の範囲のサブルーチン情報を処理概要情報として生成する処理について説明する。

【００８１】

同図に示すように、この処理概要生成部１１６は、サブルーチン情報記憶部１２２に記憶されたサブルーチン情報のツリー構造を読み出し、読み出したツリー構造から一つのサ

ブルーチン情報を取り出す（ステップＳ１４１）。そして、呼出階層値を取得し（ステップＳ１４２）、取得した呼出階層値が指定の範囲内か否かを判定する（ステップＳ１４３）。

【００８２】

その結果、呼出階層値が指定の範囲にない場合には、そのサブルーチン情報をツリー構造から削除する（ステップＳ１４４）。そして、全てのサブルーチン情報をツリー構造から取り出して呼出階層値を調べたか否かを判定し（ステップＳ１４５）、全てのサブルーチン情報を調べていない場合には、ステップＳ１４１に戻って次のサブルーチン情報を調べ、全てのサブルーチン情報を調べた場合には、残ったツリー構造のサブルーチン情報を処理概要情報記憶部１２４に格納する（ステップＳ１４６）。

【００８３】

このように、この処理概要生成部１１６がサブルーチン情報記憶部１２２に記憶されたツリー構造のサブルーチン情報から、指定された範囲のサブルーチン情報だけを取り出すことによって、プログラムの呼出構造と入出力情報とが関連づけられた処理概要を生成することができる。

【００８４】

上述してきたように、本実施例１では、構文解析部１１１による構文解析結果から構造抽出部１１２がサブルーチンの呼出構造を抽出し、入出力データ抽出部１１４が各サブルーチンの入出力情報を抽出する。そして、構文－入出力データ関連生成部１１６が条件分岐情報抽出部１１３、構造抽出部１１２および入出力データ抽出部１１４により抽出された情報に基づいてプログラムの呼出構造と入出力情報の関連づけを行い、処理概要生成部１１６が構文－入出力データ関連生成部１１６により関連づけられた情報から指定された範囲の情報を抽出して処理概要情報を作成し、ドキュメント生成部１１７が処理概要生成部１１６により生成された処理概要情報に基づいてプログラム仕様書を作成することとしたので、プログラムの処理概要の把握を容易にすることができる。

【００８５】

また、本実施例１では、構文解析部１１１による構文解析結果から条件分岐情報抽出部１１３が条件分岐の条件情報を抽出し、構造抽出部１１２がサブルーチンの条件分岐の条件情報をサブルーチン情報に加えることとしたので、どのような条件の際に入出力が行われているかの把握を容易にすることができる。

【実施例２】

【００８６】

ところで、図８に示したプログラム仕様書の［コメント欄］には、利用者は自由にコメントを追記することができる。しかしながら、利用者が追記したコメントは、新たなプログラム仕様書を自動生成した場合に、新たに生成されたプログラム仕様書には反映されないため、利用者はコメントを再度追記する必要がある。そこで、本実施例２では、新たにプログラム仕様書を自動生成した場合に、利用者によるコメントの再追記を不要とする仕様書生成装置について説明する。

【００８７】

まず、本実施例２に係るコメント継承について説明する。図１３は、本実施例２に係るコメント継承を説明するための説明図である。同図は、文書作成アプリケーションで作成されたプログラム仕様書に設けられるコメント記述領域を示す。

【００８８】

コメント記述領域は、文書作成アプリケーションによって表形式で作成されており、各コメント記述領域には、「コメント（ユーザ記述領域）」や「［変数一覧］」などのキーワードがつけられている。また、コメント記述領域には、一つの記述領域しかない場合（タイプ１）と、複数の記述領域がある場合（タイプ２）とがある。

【００８９】

本実施例２に係る仕様書生成装置は、これらのコメント記述領域に追記された利用者のコメントをプログラム仕様書から読み取り、生成するプログラム仕様書の同じ位置に読み

取ったコメントを自動的に継承する。

【００９０】

また、プログラム仕様書には補足領域を設け、コメントが追記されたコメント記述領域が新たに生成するプログラム仕様書にない場合には、補足領域にそのコメントを記入する。

【００９１】

このように、本実施例２に係る仕様書生成装置は、コメントの追記されたプログラム仕様書を読み込み、新たに生成するプログラム仕様書の同じ位置に読み取ったコメントを自動的に継承することによって、利用者がコメントとして追記した業務情報や運用ノウハウなどの重要な情報を新旧のプログラム仕様書間で継承することができる。

【００９２】

次に、本実施例２に係る仕様書生成装置の構成について説明する。図１４は、本実施例２に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図１に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【００９３】

同図に示すように、この仕様書生成装置２００は、制御部２１０と、制御部２１０による処理の途中結果などを記憶する記憶部１２０とを有する。制御部２１０は、図１に示したドキュメント生成部１１７の代わりにドキュメント生成部２１１を有する。

【００９４】

ドキュメント生成部２１１は、ドキュメント生成部１１７の有する機能に加えて、コメント記入内容継承部２１１ａを有する。このコメント記入内容継承部２１１ａは、コメント記入済プログラム仕様書を読み込んでコメントを抽出し、新たに生成するプログラム仕様書に、抽出したコメントを反映する。

【００９５】

このコメント記入内容継承部２１１ａがコメント記入済プログラム仕様書を読み込んでコメントを抽出し、新たに生成するプログラム仕様書に、抽出したコメントを反映することによって、仕様書生成装置２００は、古いプログラム仕様書に記入されたコメントを新たに生成するプログラム仕様書に継承することができる。

【００９６】

次に、コメント記入内容継承部２１１ａがコメント継承処理で使用するデータ構造について説明する。仕様書生成装置２００は、コメント記述領域の位置情報をプログラム仕様書に付加するプロパティに対応させ、これらのプロパティのリストを値とするプロパティをコメント記述領域リストとして管理する。

【００９７】

図１５は、コメント記述領域リストに対応するプロパティのデータ構造の一例を示す図である。同図に示すように、このコメント記述領域リストは、ReflectionListというプロパティであり、プロパティの値は、各コメント記述領域に対応するプロパティ「Reflection１」～「ReflectionN」から構成されるリストである。

【００９８】

図１６－１は、コメント記述領域に対応するプロパティのデータ構造の一例を示す図である。同図は、コメント記述領域に対応するプロパティの一例である「Reflection1」のデータ構造を示しており、「Reflection1」はコメント記述領域を識別するキーワードをプロパティの値としている。

【００９９】

また、図１６－２は、複数の記述領域がある場合のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。同図は、複数の記述領域がある場合のコメント記述領域に対応するプロパティの一例である「MultiReflection1」のデータ構造を示しており、「MultiReflection1」は、コメント記述領域を識別するキーワードと、それぞれの記述領域を識別する複数コメントキーワードの列の位置と、コメント記述領域の列の位置

をプロパティの値としている。

【０１００】

例えば、図１３に示したコメント記述領域［変数一覧］に対応するプロパティの値は、「［変数一覧］，１，２」となる。すなわち、コメント記述領域［変数一覧］の１列目が各記述領域を識別する複数コメント識別キーワードであり、２列目が記入されたコメントである。

【０１０１】

次に、図１４に示したコメント記入内容継承部２１１ａの処理手順について説明する。図１７は、図１４に示したコメント記入内容継承部２１１ａの処理手順を示すフローチャートである。

【０１０２】

同図に示すように、このコメント記入内容継承部２１１ａは、コメントの継承元のプログラム仕様書のReflectionListを探索し（ステップＳ２１１）、ReflectionListからReflectionを一つ取得する（ステップＳ２１２）。

【０１０３】

そして、取得したReflectionの値であるキーワードを用いて継承元のコメントと継承先のコメント継承位置とを取得し（ステップＳ２１３およびステップＳ２１４）、継承先のプログラム仕様書にコメント継承位置が存在するか否かを判定する（ステップＳ２１５）。その結果、コメント継承位置が存在する場合には、継承先のプログラム仕様書に継承元のプログラム仕様書のコメントを反映する（ステップＳ２１６）。

【０１０４】

そして、コメントの継承が成功したか否かを判定し（ステップＳ２１７）、継承が成功しなかった場合には、補足領域にコメントを反映する（ステップＳ２１８）。そして、ReflectionListの全てのReflectionを処理したか否かを判定し（ステップＳ２１９）、全てのReflectionを処理していない場合には、ステップＳ２１２に戻って次のReflectionの処理を行い、全てのReflectionを処理した場合には、処理を終了する。

【０１０５】

このように、このコメント記入内容継承部２１１ａがReflectionListを用いて継承元のプログラム仕様書からコメントを読み出して継承先のプログラム仕様書に反映させることによって、利用者がプログラム仕様書に追記したコメントを自動的に継承することができる。

【０１０６】

次に、他の形式のプログラム仕様書のコメント記述領域について説明する。まず、スプレッドシートを用いたプログラム仕様書のコメント記述領域について説明する。図１８は、スプレッドシートを用いたプログラム仕様書のコメント記述領域の一例を示す図である。

【０１０７】

同図に示すように、スプレッドシートの場合には、全ての領域が行と列で位置を特定できるため、コメント記述領域に対応するプロパティの値はキーワード、行位置および列位置となる。

【０１０８】

図１９は、スプレッドシートを用いたプログラム仕様書のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。同図は、コメント記述領域に対応するプロパティの一例である「Reflection１」のデータ構造を示しており、「Reflection１」はキーワードとコメント記述領域行位置とコメント記述領域列位置とをプロパティの値としている。

【０１０９】

ここで、コメント記述領域行位置は、キーワードからの相対行位置である。例えば、図１８に示したコメント記述領域「コメント（ユーザ記述領域）」に対応するプロパティの値は、「コメント（ユーザ記述領域），１，Ａ）」となる。

【０１１０】
また、図２０は、スプレッドシートを用いたプログラム仕様書のコメント記述領域（複数の記述領域がある場合）に対応するプロパティのデータ構造の一例を示す図である。同図は、複数の記述領域がある場合のコメント記述領域に対応するプロパティの一例である「MultiReflection１」のデータ構造を示しており、「MultiReflection１」はキーワードとコメント記述領域行位置と一コメントの行数と複数コメント識別キーワード列位置とコメント記述領域列位置とを値とをプロパティの値としている。
【０１１１】
ここで、コメント記述領域行位置は、キーワードからの相対行位置である。例えば、図１８に示したコメント記述領域「［変数一覧］」に対応するプロパティの値は、「［変数一覧］，２，１，Ａ，Ｂ」となる。
【０１１２】
図２１は、処理構造図におけるコメント記述領域を示す図である。なお、処理構造図の詳細については後述する。同図において、「コメント（ｎ）」の箇所がコメント記述領域である。
【０１１３】
この処理構造図では、自動生成した内容が入ったセルと利用者がコメントを追記可能なセルとを区別するために、前者のセルは書き込み禁止とし、両者のセルの色を変えておく。また、各セクションにコメント記入可能なセルが必ず存在するように、処理構造図生成時に各セクションの有効掛線範囲には、必ず空セルを入れるようにしておく。
【０１１４】
図２２は、図２１に示した処理構造図における各コメント記述領域の位置情報データの一例を示す図である。同図に示すように、この例では、コメント記述領域が対応するセクション構造と、対応セクションからの相対的変位と、付加列名とをコメント記述領域の位置情報としている。
【０１１５】
例えば、「コメント（１）」は「プログラム名」のセクションに対応し、位置は「プログラム名」のセルから２行０列目であり、「コメント（４）」は「プログラム名－繰り返し処理［１］－CALL"COM0002"［１］」のセクションに対応し、位置は「CALL"COM0002"」のセルから１行０列目である。
【０１１６】
仕様書生成装置２００は、このような位置情報データを用いて処理構造図のコメント記述領域を管理することによって、処理構造図に追記された利用者のコメントを自動的に継承することができる。
【０１１７】
また、仕様書生成装置２００は、継承先の処理構造図に対応するセクション構造は存在するが対応するセルや列がない場合には、備考欄にコメントを記入する。一方、継承先の処理構造図に対応するセクション構造が存在しない場合には、プログラム仕様書の末端の補足欄にコメント記述領域の位置情報とコメント内容とを記入する。
【０１１８】
また、後述する画面遷移図のようなページ制御があるようなワークシートの場合は、出力の度に位置情報が変わるため、上記の方法が使えない。そのため、１枚のシートを丸ごと利用者記入欄として設定し、利用者記入用としたシートに書いたコメントを丸ごと継承するようにする。
【０１１９】
図６５は、利用者記入欄を設けないコメント継承の一例を示す図である。同図は、利用者記入欄を設けずに、コメント記入用のインタフェースを用意し、利用者がコメントを自由に記入する場合を示している。
【０１２０】
利用者はあらかじめ用意されたインタフェース（例えばコメント記入ボタンなど）を用

いてコメントを記入する。そして、記入されたコメントは別文書として管理し、仕様書本体と関連づけておく。関連づけの方法には例えば文書の命名規則を決めておき、「ＸＸＸ」という名前の仕様書に追加されたコメント文書は「ＸＸＸ-コメント1」となるようにしておき、仕様書全体としてユニークになるようにしておけばよい。

【０１２１】

こうしておくと、仕様書を再生成しても「ＸＸＸ」は上書きされるが、「ＸＸＸ-コメント１」は上書きされないので、利用者が記入したコメントは保持されたままになる。また、再生成した仕様書から、前回のコメントを参照する場合は、コメントを表示するインタフェースを使って関連文書からコメント内容を抽出して表示する。

【０１２２】

例えば、図６５は、仕様書「Ｆ１０８０２Ｎ」へのコメントとして「修正記録０３０２１７」および「障害対応メモ」があり、「修正記録０３０２１７」の内容が「ジョブステップ３での在庫マスタの更新ロジックは’０３／２／１７修正済」であることを示している。

【０１２３】

上述してきたように、本実施例２では、ドキュメント生成部２１１のコメント記入内容継承部２１１ａが、コメント記入済のプログラム仕様書からコメントを取り出し、新たに生成するプログラム仕様書の対応する場所に反映させることとしたので、利用者によりプログラム仕様書に追記されたコメントを自動的に継承することができる。これによって、保守担当者の交替時などに、前任者の知識をコメントとしてプログラム仕様書に書いておくことで引き継ぎ作業に伴う工数を大幅に減らすことができる。

【実施例３】

【０１２４】

実施例１で説明したように、サブルーチンの呼出構造と入出力情報とを関連づけた処理概要によってプログラムの基本的な処理構造は理解することができる。しかしながら、プログラムを理解するためには、基本的な処理構造に加えてプログラムの処理内容も理解する必要があることも多い。そこで、本実施例３では、プログラムの処理内容の概要をプログラムから抽出してプログラム概要書として生成する仕様書生成装置について説明する。

【０１２５】

まず、本実施例３に係る仕様書生成装置の構成について説明する。図２３は、本実施例３に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図１に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【０１２６】

同図に示すように、この仕様書生成装置３００は、制御部３１０と、制御部３１０が使用する情報や制御部３１０による処理の途中結果などを記憶する記憶部３２０とを有する。制御部３１０は、図１に示した制御部１１０が有する機能部に加えてプログラム概要生成部３１１を有し、ドキュメント生成部１１７の代わりにドキュメント生成部３１２を有する。

【０１２７】

また、記憶部３２０は、図１に示した記憶部１２０が有する記憶部に加えて、まとめ処理記憶部３２１と、日本語テンプレート記憶部３２２と、プログラム概要情報記憶部３２３とを有する。

【０１２８】

プログラム概要生成部３１１は、プログラムから処理内容の概要をプログラム概要情報として生成する処理部であり、プログラム概要情報記憶部３２３は、プログラム概要生成部３１１が生成したプログラム概要情報を記憶する記憶部である。

【０１２９】

ドキュメント生成部３１２は、処理概要情報を用いてプログラム仕様書を生成するとともに、プログラム概要情報記憶部３２３に記憶されたプログラム概要情報を用いてプログ

ラム概要書を生成する。

【０１３０】

まとめ処理記憶部３２１は、プログラム概要生成部３１１がプログラム概要情報を生成するために行うまとめ処理の内容を記憶した記憶部であり、日本語テンプレート記憶部３２２は、プログラム概要生成部３１１が日本語によるプログラム概要を生成するための日本語テンプレートを記憶した記憶部である。

【０１３１】

図２４は、プログラム概要生成部３１１の構成を示す機能ブロック図である。同図に示すように、このプログラム概要生成部３１１は、データ重要度設定部３１１ａと、ステートメント重要度設定部３１１ｂと、まとめ管理部３１１ｃと、日本語変換部３１１ｄとを有する。

【０１３２】

データ重要度設定部３１１ａは、構文解析情報記憶部１２１に記憶された構文解析情報をＸＭＬ形式のプログラム中間情報に変換し、プログラム中間情報に含まれるデータに対して重要度を割り当てる処理部である。

【０１３３】

図２５－１および図２５－２は、プログラム中間情報を説明するための図であり、図２５－１は、プログラムソース例を示す図であり、図２５－２は、図２５－１に示したプログラムソース例から生成されるプログラム中間情報を示す図である。

【０１３４】

図２５－２に示すように、プログラム中間情報では、基本的にプログラムのステートメント（文）一つにつき一つのタグが対応している。また、パラメータ等はステートメントの内部に入れ子のタグを用いて記述する。

【０１３５】

例えば、「ＭＯＶＥ　０　ＴＯ　Ｗ－ＥＲＲ－ＦＬＡＧ」というステートメントは、

＜ｍｏｖｅ＞
　＜ｒｅｆ＞＜ｃｏｎｓｔａｎｔ　ｖａｌｕｅ＝”０”　ｔｙｐｅ＝”ｉｎｔ”／＞＜／ｒｅｆ＞
　＜ｄｅｆ＞＜ｖａｒ　ｎａｍｅ＝”Ｗ－ＥＲＲ－ＦＬＡＧ”／＞＜／ｄｅｆ＞
＜／ｍｏｖｅ＞

というプログラム中間情報に変換される。

【０１３６】

図２６は、データの重要度の分類の一例を示す図である。同図に示すように、データ重要度設定部３１１ａは、「処理パスの分岐条件に関わるデータ」を重要度が最も高い重要度「１」に分類して分類ＩＤを「Ｄ－１」とし、「ファイル／データベースなどの出力に使われるデータ」を重要度「２」に分類して分類ＩＤを「Ｄ－２」とする。

【０１３７】

また、データ重要度設定部３１１ａは、「ファイル／データベースなどからの入力に使われるデータ」を重要度「３」に分類して分類ＩＤを「Ｄ－３」とし、「その他のデータ」を重要度が最も低い重要度「４」に分類して分類ＩＤを「Ｄ－４」とする。

【０１３８】

そして、データ重要度設定部３１１ａは、プログラム中の全ての変数に対して、プログラム中間情報の変数用タグに属性の形でデータの重要度を付加する。図２７は、プログラム中間情報の変数用タグに付加される重要度の一例を示す図である。

【０１３９】

同図に示すように、変数「ＤＡＴＡ１」は、プログラム中間情報では＜ｖａｒ　ｎａｍｅ＝”ＤＡＴＡ１”／＞と表わされ、重要度「１」が付加されると、＜ｖａｒ　ｎａｍｅ＝”ＤＡＴＡ１”　ｄａｔａ＿ｐｒｉｏｒｉｔｙ＝”１”／＞と表わされる。

【０１４０】

ステートメント重要度設定部３１１ｂは、データの重要度が付加されたプログラム中間

情報にステートメントの重要度を付加する処理部である。図２８は、ステートメントの重要度の分類の一例を示す図である。

【０１４１】

同図に示すように、ステートメント重要度設定部３１１ｂは、「重要度がＤ－１およびＤ－２に該当するデータに対し代入等の書き換えを行う文、サブルーチン呼出を行う文またはファイル、データベースに入出力を行う文」を重要度が最も高い重要度「１」に分類して分類ＩＤを「Ｓ－１」とし、「条件判定文のうち判定直後に実行される文としてＳ－１の分類の文を含むもの」を重要度「２」に分類して分類ＩＤを「Ｓ－２」とし、「その他の文」を重要度「３」に分類して分類ＩＤを「Ｓ－３」とする。

【０１４２】

そして、ステートメント重要度設定部３１１ｂは、プログラム中間情報に記述されているステートメントのタグとデータの重要度を分析し、各ステートメントの重要度を決定する。そして、決定したステートメントの重要度をタグの属性として付加する。

【０１４３】

図２９は、プログラム中間情報のステートメントのタグに付加される重要度の一例を示す図である。同図に示すように、「ＭＯＶＥ　０　ＴＯ　ＤＡＴＡ１」というステートメントについては、変数「ＤＡＴＡ１」に重要度「１」が付加されているので、＜ｍｏｖｅ　ｓｔａｔｅｍｅｎｔ＿ｐｒｉｏｒｉｔｙ＝”１”＞のようにステートメントの重要度「１」が付加される。

【０１４４】

このようにして、ステートメント重要度設定部３１１ｂは、ステートメントの重要度を付加したプログラム中間情報を生成する。なお、データの重要度およびステートメントの重要度の分類については、まとめ管理部３１１ｃの処理で必要に応じて変更されたり細分化される場合もある。

【０１４５】

まとめ管理部３１１ｃは、データとステートメントの重要度が付加されたプログラム中間情報について、様々なまとめ処理を実行する処理部である。このまとめ管理部３１１ｃが行うまとめ処理には、プログラムの局所部分についてまとめを行うものと、プログラムの全体的な処理の流れについてまとめを行うものがある。

【０１４６】

局所的なまとめ処理では、プログラムの一部分についてのパターンを抽出し、抽出したパターンに対して所定の処理を行う。例えば、データ項目の代入処理が連続してプログラムに記述されており、そのデータ項目はある同じ集団項目に属している場合、その集団項目を用いた代入という形で代入処理を一つにまとめてしまう。

【０１４７】

また、局所的なまとめ処理では、重要でない文の削除も行う。あるいは、構造抽出部１１２によって抽出されたサブルーチン呼出階層値を用いて、呼び出し構造的に上位のサブルーチンを残すといった処理も行う。

【０１４８】

全体的なまとめ処理では、例えば、ジャンプ文によりプログラム全体が一つの大きなループになっていることを検出し、それをループしているという表現で置き換えるといった処理を行う。

【０１４９】

具体的には、このまとめ管理部３１１ｃは、まとめ処理を実行するまとめ処理プログラムを最初に読み込み、読み込んだまとめ処理プログラムを実行することによって、まとめ処理を実行する。

【０１５０】

図３０は、まとめ管理部３１１ｃの初期動作を説明するための説明図である。同図に示すように、まとめ管理部３１１ｃは、まとめ処理プログラムのプログラム名を、まとめ登録情報として用意し、このまとめ登録情報に登録されたまとめ処理プログラムを最初に読

み込んでおく。

【０１５１】

同図では、読み込むプログラム名として「ＡＲＲＡＮＧＥ＿Ｐ１」、「ＡＲＲＡＮＧＥ＿Ｐ２」、「ＡＲＲＡＮＧＥ＿Ａ１」および「ＡＲＲＡＮＧＥ＿Ａ２」が登録されており、まとめ管理部３１１ｃは、これらのまとめ処理プログラムを読み込んで実行する。

【０１５２】

なお、「ＡＲＲＡＮＧＥ＿Ｐ１」および「ＡＲＲＡＮＧＥ＿Ｐ２」は局所的なまとめ処理を行うプログラムであり、「ＡＲＲＡＮＧＥ＿Ａ１」および「ＡＲＲＡＮＧＥ＿Ａ２」は局所的なまとめ処理を行うプログラムである。また、まとめ登録情報に登録するプログラム名を入れ替えることによって、行いたいまとめ処理を容易に変更することができる。

【０１５３】

図３１は、まとめ管理部３１１ｃとまとめ処理プログラムとのインタフェース（Ｉ／Ｆ）を示す図である。同図に示すように、インタフェースとしては、処理名取得Ｉ／Ｆと、種類取得Ｉ／Ｆと、優先度番号取得Ｉ／Ｆと、処理実行Ｉ／Ｆとがある。

【０１５４】

処理名取得Ｉ／Ｆは、まとめ処理プログラムが行うまとめ処理の処理名（例えば”集団項目のまとめ”）をまとめ管理部３１１ｃに通知するためのインタフェースであり、種類取得Ｉ／Ｆは、まとめ処理の種類（例えば”局所”、”全体”）をまとめ管理部３１１ｃに通知するためのインタフェースである。

【０１５５】

また、優先度番号取得Ｉ／Ｆは、まとめ処理プログラムの優先度番号（例えば”１０”、”１０４”）をまとめ管理部３１１ｃに通知するためのインタフェースであり、処理実行Ｉ／Ｆは、まとめ管理部３１１ｃがまとめ処理プログラムを実行するときに使用するインタフェースである。

【０１５６】

なお、まとめ管理部３１１ｃがまとめ処理プログラムを実行するときに渡すパラメータは、主にプログラム中間情報である。また、まとめ処理の中に構造抽出部１１２によって抽出された構造情報などを必要とするものがあれば、それらの情報もパラメータとして渡す。

【０１５７】

また、優先度番号は、まとめ処理プログラムの処理順を決めるための相対的な番号であり、ここでは、この優先度番号の若い順にまとめ処理プログラムを実行する。まとめ管理部３１１ｃは、インタフェースを介して取得した情報を用いて、まとめ処理プログラムの実行順序を決定して保持する。

【０１５８】

図３２は、まとめ処理プログラムの実行順序保持の一例を示す図である。同図は、局所的なまとめ処理を行うまとめ処理プログラムを先に実行し、全体的なまとめ処理を行うまとめ処理プログラムを後に実行し、それぞれの処理では、優先番号の小さい順に実行する順序保持を示している。

【０１５９】

なお、まとめ処理の処理名、種類、優先度番号は、まとめ処理プログラム開発時に開発者によって定義される。また、優先度番号が同じ場合には、まとめ登録情報に登録された順番に実行順序の保持がなされる。

【０１６０】

まとめ管理部３１１ｃは、データとステートメントの重要度が付加されたプログラム中間情報を入力し、保持するまとめ処理プログラムの実行順序に従って、まとめ処理プログラムを実行していく。

【０１６１】

また、まとめ管理部３１１ｃは、まとめ処理プログラムにプログラム中間情報を入力として与え、各まとめ処理プログラムは、プログラム中間情報を編集していく。そして、各

まとめ処理プログラムで編集されたプログラム中間情報はその次に実行されるまとめ処理プログラムの入力となる。

【０１６２】

このようにして、登録されているすべてのまとめ処理プログラムによって、プログラム中間情報の編集が行われていく。なお、まとめ処理プログラムの中には、プログラム中間情報から自分が処理できるプログラム構造のパターンを抽出するものがあるが、パターンが抽出できなかった場合は、そのまとめ処理を行わず、次のまとめ処理プログラムへ実行が引き渡される。そして、まとめ管理部３１１ｃは、すべてのまとめ処理プログラムの実行を終了すると、編集したプログラム中間情報を日本語変換部３１１ｄに引き渡す。

【０１６３】

図３３－１～図３３－６は、まとめ処理プログラムが行うまとめのルールの例を示す図である。図３３－１～図３３－４は、局所的なまとめのルールを示し、図３３－５および図３３－６は、全体的なまとめのルールを示す。

【０１６４】

図３３－１は、「ＲＥＡＤ文のファイル終了時の処理が、ＵＮＴＩＬ条件の成立と一致している場合、ＵＮＴＩＬの条件を「ファイル～の終わりまで」に置き換える。また、ＲＥＡＤ文のファイル終了時の処理を削除する。」というまとめのルールを示す。

【０１６５】

図３３－２は、「セクションの内容が短く、３行までの場合でかつ、呼出元（上位セクション）にて無条件で呼ばれている場合、セクションの内容を上位のセクションに含む形に展開する。」というまとめのルールを示す。

【０１６６】

図３３－３は、「ＭＯＶＥ文が連続し、代入元の項目がすべて同じ集団項目に所属し、また、代入先の項目がすべて同じ集団項目に所属するとき、それら集団項目による代入文に置き換える。なお、集団項目内のすべての項目が列挙されていない場合は、その集団項目の部分であることを示す属性を加える。」というまとめのルールを示す。

【０１６７】

図３３－４は、「（１）ｓｔａｔｅｍｅｎｔ＿ｐｒｉｏｒｉｔｙが”３”になっている文を削除する。また、このとき、削除される文のタグの親タグ＜ｓｅｑｕｅｎｃｅ＞の属性qtの値を削除した文の数だけ減らす（ｑｔはその内部に含む文の数を示す）。（２）また、このとき、＜ｓｅｑｕｅｎｃｅ＞の属性ｑｔの値が０の場合、それを含むセクション、パラグラフまたは、条件文（ＩＦ、ＥＶＡＬＵＡＴＥ、ＲＥＡＤ後条件）を削除する。セクションの場合は、その呼出元の呼出文（ＥＶＡＬＵＡＴＥ）も削除する。なお、（１）、（２）は、文の数が変わらなくなるまで対象となるプログラムについて繰り返し行う。」というまとめのルールを示す。

【０１６８】

図３３－５は、「ファイルを読む処理があり、その後にループがあり、ループの内部の処理の最後で、ファイルを読んでいる場合、「ループの条件成立まで、繰り返しファイルを読む」という形に置き換える。ループがセクションの場合は、それを呼び出している部分に展開する。」というまとめのルールを示す。

【０１６９】

図３３－６は、「変数の項目名を集団項目名も付加した形で表現する。名前の間は、”．”で連結する。また、もっとも上位の集団項目から表現する。例えば、ＩＮ－ＲＥＣＯＲＤ．ＩＮ－ＤＡＴＡ１などの形式。」というまとめのルールを示す。

【０１７０】

日本語変換部３１１ｄは、まとめ処理で編集されたプログラム中間情報を受け取り、各タグを対応する日本語テンプレートに対応させて日本語情報を生成する処理部である。この日本語変換部３１１ｄによって生成された日本語情報は、プログラム概要記憶部３２３に格納され、ドキュメント生成部３１２によって、プログラム概要書に記載される。

【０１７１】

図３４は、日本語変換部３１１ｄの処理を説明するための説明図である。同図は、”ＭＯＶＥ　０　ＴＯ　ＤＡＴＡ１”というＭＯＶＥ文を「整数（０）を変数［ＤＡＴＡ１］に代入する。」という日本語情報に変換する例を示している。

【０１７２】

ここで、日本語変換部３１１ｄは、「＜ｍｏｖｅ＞＜ｒｅｆ＞～（１）＜／ｒｅｆ＞＜ｄｅｆ＞～（２）＜／ｄｅｆ＞＜／ｍｏｖｅ＞」→「～（１）を～（２）に代入する。」という日本語テンプレートと、「＜ｃｏｎｓｔａｎｔ　ｖａｌｕｅ＝”～（３）”　ｔｙｐｅ＝”ｉｎｔ”／＞」→「整数（～３））」という日本語テンプレートと、「＜ｖａｒ　ｎａｍｅ＝”～（４）”／＞→「変数［～（４）］」という日本語テンプレートを用いて変換を行っている。また、図３５－１および図３５－２は、その他の日本語テンプレートの例を示す図である。

【０１７３】

次に、プログラム概要生成部３１１が生成するプログラム概要の例について説明する。図３６は、プログラムサンプルを示す図である。プログラム概要生成部３１１は、同図に示したプログラムサンプルの構文解析情報をプログラム中間情報に変換し、データおよびステートメントに重要度を付加する。

【０１７４】

例えば、図３６の［１］で示される変数ＥＮＤ－ＦＬＡＧは、セクション繰り返し処理の終了条件であり、処理の分岐であるので、重要度は「１」となり、図３６の［２］で示される変数ＩＮ－ＣＯＵＮＴは、「Ｄ－１」から「Ｄ－３」の分類に該当しないので、重要度は「４」となる。また、図３６の［３］で示されるＷＲＩＴＥ文は、ファイル入出力に関わるので、重要度は「１」となる。

【０１７５】

そして、プログラム概要生成部３１１のまとめ管理部３１１ｃが、データおよびステートメントに重要度が付加されたプログラム中間情報に対して図３５－１～図３５－６に示したまとめのルールを適用してまとめ処理を行う。

【０１７６】

図３７は、まとめ管理部３１１ｃが利用するまとめ処理プログラムのリストを示す図である。まとめ管理部３１１ｃは、図３１に示したインタフェースを用いてまとめ処理プログラムから図３７に示す情報を取得する。

【０１７７】

また、図３８は、まとめ処理プログラムの実行順序を示す図である。まとめ管理部３１１ｃは、図３７に示したまとめ処理プログラムのリストに基づいて、図３８に示す実行順序でまとめ処理プログラムを実行する。

【０１７８】

すなわち、まとめ管理部３１１ｃは、局所的な処理としてＡＲ＿Ｐ１、ＡＲ＿Ｐ２、ＡＲ＿Ｐ３およびＡＲ＿Ｐ４を順に実行し、その後、全体的な処理としてＡＲ＿Ａ１およびＡＲ＿Ａ２を実行する。

【０１７９】

図３９は、まとめ処理終了時のプログラム中間情報を示す図である。日本語変換部３１１ｄは、図３９に示すプログラム中間情報に対して図３４、図３５－１および図３５－２に示した日本語テンプレートを適用し、図４０に示す日本語情報を生成する。

【０１８０】

また、ドキュメント生成部３１２は、図４０に示す日本語情報を用いて、図４１に示すようなプログラム概要書を生成する。

【０１８１】

上述してきたように、本実施例３では、プログラム概要生成部３１１が、構文解析部１１１により生成された構文解析情報をプログラム中間情報に変換し、まとめ処理記憶部３２１に記憶されたまとめ処理プログラムと日本語テンプレート記憶部３２２に記憶された日本語テンプレートとを用いてプログラム中間情報からプログラム概要を生成することと

したので、プログラムの処理内容の理解を容易し、理解に要する時間を短縮することができる。

【実施例４】

【０１８２】

ところで、実施例１～実施例３では、一つのプログラムを解析してその処理概要をプログラム仕様書として生成する仕様書生成装置について説明したが、バッチ処理システムでは、複数のプログラムを順に実行することによって所定の処理を行うことも多い。

【０１８３】

このような複数のプログラムを実行するバッチ処理システムでは、各プログラムの処理概要を理解することも重要であるが、バッチ処理全体としての処理概要を理解することが重要となる。

【０１８４】

そこで、本実施例４では、バッチ処理全体の処理の概要を抽出して仕様書を生成する仕様書生成装置について説明する。具体的には、本実施例４に係る仕様書生成装置は、バッチ処理全体における入力データと出力データ、および主要なプログラムを抽出し、バッチジョブ全体の処理概要を示す仕様書を作成する。

【０１８５】

まず、本実施例４に係る仕様書生成装置の構成について説明する。図４２は、本実施例４に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図１に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【０１８６】

同図に示すように、この仕様書生成装置４００の制御部４１０は、図１に示した制御部１１０が有する機能部に加えてバッチジョブ記述言語構文解析部４１１と、ジョブステップ入出力データ抽出部４１２と、全体入出力情報抽出部４１３と、主要プログラム判定部４１４とを有する。また、処理概要生成部１１６の代わりに処理概要生成部４１５を有し、ドキュメント生成部１１７の代わりにドキュメント生成部４１６を有する。

【０１８７】

バッチジョブ記述言語構文解析部４１１は、バッチジョブ記述言語ソースを読み込み、各ジョブステップにおける呼出プログラム名や使用ファイル名の情報を抽出したバッチジョブ解析情報を生成する処理部である。

【０１８８】

ジョブステップ入出力データ抽出部４１２は、バッチジョブ記述言語構文解析部４１１が生成したバッチジョブ解析情報から、各ジョブステップごとに、ファイルが入力に使用されているか、出力に使用されているか、あるいは削除されているかを判定し、判定した結果をジョブステップ入出力情報として生成する処理部である。

【０１８９】

全体入出力情報抽出部４１３は、ジョブステップ入出力データ抽出部４１２が生成したジョブステップ入出力情報と、入出力データ抽出部１１４が生成した入出力情報リストとを用いて、ジョブ全体として入力に使用されるファイルと出力に使用されるファイルを決定する処理部である。

【０１９０】

主要プログラム判定部４１４は、全体入出力情報抽出部４１３が決定したジョブ全体としての入出力ファイルを使用するジョブステップを抽出し、抽出したジョブステップで呼び出しているプログラムの一覧を生成する処理部である。

【０１９１】

また、この主要プログラム判定部４１４は、抽出したジョブステップで呼び出しているプログラムの一覧を生成する場合に、呼び出しているプログラムがユーティリティプログラムである場合には、プログラムの一覧から除外する。

【０１９２】

処理概要生成部４１５は、バッチジョブに含まれる各プログラムの処理概要情報を生成するとともに、全体入出力情報抽出部４１３が決定した入出力ファイルと、主要プログラム判定部４１４が生成したプログラムの一覧とを組み合わせてバッチジョブ処理概要を生成する処理部である。

【０１９３】

このように、この処理概要生成部４１５が、各処理プログラムの処理概要情報に加えて、全体入出力情報抽出部４１３が決定した入出力ファイルと主要プログラム判定部４１４が生成したプログラムの一覧とを用いてバッチジョブ処理概要を生成することによって、仕様書生成装置４００は、プログラム仕様書に加えてジョブ処理全体の処理概要書を作成することができる。

【０１９４】

ドキュメント生成部４１６は、処理概要生成部４１５が生成した処理概要情報を用いてプログラム仕様書を生成するとともに、バッチジョブ処理概要を所定の形式に整形し、ジョブ全体の処理概要書を生成する処理部である。

【０１９５】

次に、ジョブステップ入出力データ抽出部４１２による入出力判定処理の処理手順について説明する。図４３は、ジョブステップ入出力データ抽出部４１２による入出力判定処理の処理手順を示すフローチャートである。

【０１９６】

同図に示すように、このジョブステップ入出力データ抽出部４１２は、バッチジョブ記述言語構文解析部４１１が生成したバッチジョブ解析情報からジョブステップのジョブ記述解析情報を作成する（ステップＳ４１１）。

【０１９７】

そして、ジョブステップのジョブ記述解析情報を解析してファイルが新規作成または追記で割り当てられているか否かを調べ（ステップＳ４１２）、ファイルが新規作成または追記で割り当てられている場合には、そのファイルを出力ファイルと判定する（ステップＳ４１ａ）。

【０１９８】

一方、ファイルが新規作成または追記で割り当てられていない場合には、ジョブステップの終了時にファイルが削除されているか否かを調べ（ステップＳ４１３）、ファイルが削除されている場合には、そのファイルを削除されるファイルと判定する（ステップＳ４１４）。

【０１９９】

そして、ジョブステップで呼び出されるプログラムの構文解析情報を取得し（ステップＳ４１５）、ファイルに対して書き込みを行う文があるか否かを調べ（ステップＳ４１６）、書き込みを行う文がある場合には、そのファイルを出力ファイルと判定する（ステップＳ４１ａ）。

【０２００】

一方、ファイルに対して書き込みを行う文がない場合には、ファイルに対して読みこみを行う文があるか否かを調べ（ステップＳ４１７）、読み込みを行う文がある場合には、そのファイルを入力ファイルと判定し（ステップＳ４１８）、読み込みを行う文がない場合には、そのファイルを入出力なしと判定する（ステップＳ４１９）。

【０２０１】

このように、このジョブステップ入出力データ抽出部４１２は、ジョブ記述解析情報およびプログラム構文解析情報を用いてファイルが入力に使用されているのか出力に使用されているのかなどを判定することによって、各ジョブステップでのジョブステップ入出力情報を生成することができる。

【０２０２】

次に、全体入出力情報抽出部４１３によるジョブ全体としてのファイル入出力判定処理の処理手順について説明する。図４４は、全体入出力情報抽出部４１３によるジョブ全体

としてのファイル入出力判定処理の処理手順を示すフローチャートである。

【０２０３】

同図に示すように、この全体入出力情報抽出部４１３は、まず対象ジョブステップを先頭のジョブステップとし（ステップＳ４２１）、判定の対象ファイルをそのジョブステップで読んでいるか否かを調べる（ステップＳ４２２）。

【０２０４】

その結果、そのジョブステップで対象ファイルを読んでいる場合には、そのファイルが参照済か否かを調べ（ステップＳ４２３）、参照済でない場合には、そのファイルを入力ファイルとし、かつそのファイルを参照済とする（ステップＳ４２４）。

【０２０５】

そして、対象ファイルをそのジョブステップで書き込んでいるか否かを調べ（ステップＳ４２５）、そのジョブステップで対象ファイルを書き込んでいる場合には、そのファイルを出力ファイルとし、かつそのファイルを参照済とする（ステップＳ４２６）。

【０２０６】

そして、対象ファイルをそのジョブステップで削除しているか否かを調べ（ステップＳ４２７）、そのジョブステップで対象ファイルを削除している場合には、そのファイルが出力ファイルになっていれば消し、かつそのファイルを参照済とする（ステップＳ４２８）。

【０２０７】

そして、ジョブステップが最終ジョブステップであるか否かを調べ（ステップＳ４２９）、最終ジョブステップでない場合には、判定の対象ジョブステップを次のジョブステップとし（ステップＳ４２ａ）、ステップＳ４２２に戻り、最終ジョブステップである場合には、処理を終了する。

【０２０８】

このように、この全体入出力情報抽出部４１３がバッチジョブ全体としてのファイル入出力を判定することによって、仕様書生成装置４００は、バッチジョブ全体の入力ファイルと出力ファイルを記載した仕様書を作成することができる。

【０２０９】

次に、主要プログラム判定部４１４によるユーティリティ除外処理の処理手順について説明する。ここで、ユーティリティ除外処理とは、主要プログラム判定部４１４が主要プログラムを抽出する場合に、ユーティリティと呼ばれる一般的なプログラムを除外してプログラムを抽出するための処理である。

【０２１０】

図４５は、ユーティリティ除外処理の処理手順を示すフローチャートである。同図に示すように、このユーティリティ除外処理は、まず、全体としての入力ファイルを読み込んでいるジョブステップを対象ジョブステップとする（ステップＳ４４１）。

【０２１１】

そして、入力ファイルを読んでいるジョブステップはユーティリティであるか否かを調べ（ステップＳ４４２）、入力ファイルを読んでいるジョブステップがユーティリティでない場合には、そのジョブステップが呼び出しているプログラムを主要プログラムとする（ステップＳ４４ａ）。

【０２１２】

また、入力ファイルを読んでいるジョブステップがユーティリティである場合には、そのジョブステップの出力ファイルは一つだけであるか否かを調べ（ステップＳ４４３）、一つだけでない場合には、ユーティリティ除外処理の結果を空とする（ステップＳ４４９）。

【０２１３】

一方、そのジョブステップの出力ファイルが一つだけである場合には、そのジョブステップからそのファイルを削除するジョブステップもしくは終了ジョブステップまでの間で、そのファイルを入力しているジョブステップを対象のジョブステップとし（ステップＳ

４４４）、対象となるジョブステップがあるか否かを調べ（ステップＳ４４５）、対象となるジョブステップがない場合には、ユーティリティ除外処理の結果を空とする（ステップＳ４４９）。

【０２１４】

これに対して、対象となるジョブステップがある場合には、それぞれのジョブステップを対象としたユーティリティ除外処理を行い（ステップＳ４４６）、全ての対象ジョブステップで主要プログラムが得られなかったかを調べ（ステップＳ４４７）、全ての対象ジョブステップで主要プログラムが得られなかった場合には、ユーティリティ除外処理の結果を空とし（ステップＳ４４９）、主要プログラムが得られたジョブステップがある場合には、全ての対象ジョブステップでの処理結果をまとめて主要プログラムとする（ステップＳ４４８）。

【０２１５】

このように、主要プログラム判定部４１４がユーティリティ除外処理によってユーティリティを除外することによって、仕様書生成装置４００は、バッチジョブの主要プログラムからユーティリティプログラムを取り除くことができる。

【０２１６】

次に、仕様書生成装置４００が生成するバッチジョブ全体の処理概要の例について説明する。図４６は、バッチジョブ記述の一例を示す図である。同図は四つのジョブステップ「ＳＴＥＰ１」～「ＳＴＥＰ４」から構成されるバッチジョブを示している。

【０２１７】

この例では、まず最初のジョブステップ「ＳＴＥＰ１」でプログラム「ＰＲＯＧＡ」を呼び出す。このとき、「ＰＲＯＧＡ」が使用するファイルとして、外部名「ＩＮ０１」にファイル「ＤＥＮＰＹＯ１」を、外部名「ＯＴ０１」にファイル「ＷＯＲＫ１」を割り当てている。

【０２１８】

これは、プログラム「ＰＲＯＧＡ」内で、「ＩＮ０１」、「ＯＴ０１」という名前で参照している入出力先に、それぞれ「ＤＥＮＰＹＯ１」と「ＷＯＲＫ１」を割り当てるということを意味している。ファイルのアクセス方法はＤＩＳＰと呼ばれるパラメータで記述しており、ＳＨＲなら既存のファイルの読み書き、（ＮＥＷ，ＰＡＳＳ）なら新規作成を意味している。

【０２１９】

また、ジョブステップＳＴＥＰ２～ＳＴＥＰ４でも同様のファイルの割当を行っており、ＳＴＥＰ２およびＳＴＥＰ３のＤＩＳＰパラメータの（ＯＬＤ，ＤＥＬＥＴＥ）は、既存ファイルの読み込みと終了時の削除を意味している。

【０２２０】

図４７は、図４６に示したバッチジョブ記述からバッチジョブ記述言語構文解析部４１１が生成するバッチジョブ解析情報を示す図である。ジョブステップ入出力データ抽出部４１２は、このバッチジョブ解析情報を用い各ジョブステップで使用する入出力ファイルを判定する。

【０２２１】

図４８は、図４６に示したバッチジョブ記述から生成されるバッチジョブ全体の処理概要を示す図である。同図に示すように、仕様書生成装置４００は、バッチ処理全体の入力ファイルとして「ＤＥＮＰＹＯ１」を抽出し、出力ファイルとして「ＳＹＵＫＥＩ１」を抽出している。

【０２２２】

また、それらのファイルを扱う処理を主要なプログラムとしてＰＲＯＧＡおよびＰＲＯＧＢを抽出している。これらの情報を出力することで、このバッチ処理が全体としてどんなデータを扱い、何を行っているのかを明確にすることができる。また、主要なプログラムについては、処理概要情報記憶部１２２に記憶された処理概要情報を用いて、その処理概要を付加することもできる。

【０２２３】
上述してきたように、本実施例４では、バッチジョブ記述言語構文解析部４１１がバッチジョブ記述言語ソースからバッチジョブ解析情報を生成し、ジョブステップ入出力データ抽出部４１２がバッチジョブ記述言語構文解析部４１１により生成されたバッチジョブ解析情報からジョブステップ入出力情報を生成し、全体入出力情報抽出部４１３がジョブステップ入出力データ抽出部４１２により生成されたジョブステップ入出力情報と、入出力データ抽出部１１４が生成した入出力情報リストとを用いて、ジョブ全体として入出力ファイルを決定し、主要プログラム判定部４１４が全体入出力情報抽出部４１３により決定されたジョブ全体としての入出力ファイルを使用するジョブステップで呼び出しているプログラムの一覧を生成することとしたので、バッチジョブ処理全体の概要の把握を容易にすることができる。
【実施例５】
【０２２４】
オンラインシステムを対象として概要を把握するためのドキュメントとしては、画面遷移図が有効である。そこで、本実施例５では、オンラインシステムを対象として画面遷移図を自動生成する仕様書生成装置について説明する。
【０２２５】
まず、本実施例５に係る仕様書生成装置の構成について説明する。図４９は、本実施例５に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図１に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。
【０２２６】
同図に示すように、この仕様書生成装置５００は、制御部５１０と、制御部５１０が使用する情報や制御部５１０による処理の途中結果などを記憶する記憶部５２０とを有する。制御部５１０は、図１に示した制御部１１０が有する機能部に加えて画面定義体解析部５１１と、画面遷移データ生成部５１２とを有し、ドキュメント生成部１１７の代わりにドキュメント生成部５１３を有する。
【０２２７】
また、記憶部５２０は、図１に示した記憶部１２０が有する記憶部に加えて、画面遷移情報記憶部５２１と、レイアウト情報記憶部５２２と、画面遷移データ記憶部５２３とを有する。
【０２２８】
画面定義体解析部５１１は、画面定義体を読み込み、各画面のレイアウト情報と、画面間の遷移条件を抽出した振り分け定義情報とを生成し、また、振り分け定義情報から、どの画面からどの画面またはプログラム呼出に遷移し、そのときの遷移条件は何かといった画面遷移情報を生成する処理部である。
【０２２９】
画面遷移情報記憶部５２１は、画面定義体解析部５１１により生成された画面遷移情報を記憶する記憶部であり、レイアウト情報記憶部５２２は、画面定義体解析部５１１により生成されたレイアウト情報を記憶する記憶部である。
【０２３０】
図５０は、画面定義体の一例を示す図である。同図に示すように、レイアウト情報は、「ＬＡＹＯＵＴ」と「ＬＡＹＥＮＤ」の間に定義された情報から生成され、画面遷移情報は、ＤＩＳＴ文によって定義された情報から生成される。
【０２３１】
図５１は、画面遷移情報の一例を示す図である。同図に示すように、この画面遷移情報は、遷移元画面と、遷移条件と、遷移先画面と、遷移先種別とを対応させた情報である。この例では、遷移元画面「Ｍ０５０００」は、「選択処理＝１」の遷移条件のもとに「Ｍ０５０１０」の画面に遷移し、「選択処理＝２」の遷移条件のもとに「ＰＧ０００１」のプログラムに遷移する。

【０２３２】
画面遷移データ生成部５１２は、画面定義体解析部５１１により生成された画面遷移情報と、処理概要生成部１１６により生成された処理概要情報とを用いて、特定の画面を起点として画面遷移図全体を構造化した画面遷移データを生成する処理部である。
【０２３３】
すなわち、この画面遷移データ生成部５１２は、画面遷移情報をもとにして、特定の画面を起点として遷移元から遷移先へと順に繋いで、全体の遷移情報を階層構造の形で表した画面遷移データを作成する。
【０２３４】
例えば、「画面Ａから画面Ｂ」への画面遷移情報と、「画面ＢからプログラムＣ」への画面遷移情報があったとすると、画面Ｂが共通部分になり、「画面Ａ→画面Ｂ→プログラムＣ」という一連の流れができる。これをすべての画面遷移情報に対して行うことによって、ひとつの巨大な画面遷移データを生成することができる。
【０２３５】
また、この画面遷移データ生成部５１２は、プログラムから生成した処理概要情報からも個々のプログラム概要文と各プログラムからの入出力ファイルデータを抽出したデータを作成し、先の画面遷移データとマージし、図５２に示すような画面遷移データを生成する。例えば、プログラムＣからファイルＸへの書き込みがある場合、先の画面遷移データと合わせて「画面Ａ→画面Ｂ→プログラムＣ→ファイルＸ」という流れが出来上がる。
【０２３６】
画面遷移データ記憶部５２３は、画面遷移データ生成部５１２により生成された画面遷移データを記憶する記憶部である。
【０２３７】
ドキュメント生成部５１３は、処理概要情報からプログラム仕様書を生成するとともに、画面遷移図データを画面遷移図として所定の形式に変換して画面遷移図書を生成する。
【０２３８】
図５３－１は、画面遷移図の一例を示す図である。同図では、例えば「画面Ｍ０５０００→画面Ｍ０５０１０→プログラムＭ０５０４０→ＭＯＤパラメータＤＢ」といった流れを示してしる。また、図５３－２は、画面遷移図の他の例を示す図であり、この例では、画面遷移図にプログラム概要が埋め込まれている。
【０２３９】
また、このドキュメント生成部５１３は、出力設定情報として指定された画面数／頁などの情報を用いてページ制御を行う。ここで、１ページに収まらなかった遷移先は、遷移先が何ページのどこにあるかという位置情報を遷移図の終端に記述しておくことで、遷移先が人目でわかるようにする。または、ページ制御をまったく行わずに１ページあたりに収容可能な画面数を指定せずに、ひとつの大きな画面遷移図を生成することも可能である。
【０２４０】
さらに、画面遷移図生成時に、画面遷移図中の各画面から別に生成していた画面レイアウト図へのリンクをはることによって、全体の流れと各画面の詳細を関連づけておくことも可能である。
【０２４１】
また、このドキュメント生成部５１３は、画面定義体解析部５１１により生成されたレイアウト情報から作成される画面レイアウト図も仕様書として出力することもできる。
【０２４２】
上述してきたように、本実施例５では、画面定義体解析部５１１が画面定義体から画面遷移情報を生成し、画面遷移データ生成部５１２が画面定義体解析部５１１により生成された画面遷移情報と処理概要生成部１１６により生成された処理概要情報とを用いて、特定の画面を起点として画面遷移図全体を構造化した画面遷移データを生成し、ドキュメント生成部５１３が画面遷移データ生成部により生成された画面遷移図データを所定の形式

に変換して画面遷移図書を生成することとしたので、オンラインシステムの処理概要の把握を容易にすることができる。

【実施例６】

【０２４３】

プログラムの処理概要を把握するためには、実施例１で説明した仕様書生成装置１００が生成するプログラム仕様書が有効であるが、プログラムの処理概要を把握した後に、プログラムの処理構造の詳細を把握する必要がある場合も多い。そこで、本実施例６では、プログラムの呼出構造と入出力情報の関係の詳細を処理構造図として生成する仕様書生成装置について説明する。

【０２４４】

まず、本実施例６に係る仕様書生成装置の構成について説明する。図５４は、本実施例６に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図１に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【０２４５】

同図に示すように、この仕様書生成装置６００は、制御部６１０と、制御部６１０による処理の途中結果などを記憶する記憶部６２０とを有する。制御部６１０は、図１に示した制御部１１０が有する機能部に加えて辞書情報合成部６１１と、処理構造生成部６１２とを有し、ドキュメント生成部１１７の代わりにドキュメント生成部６１３を有する。

【０２４６】

記憶部６２０は、図１に示した記憶部１２０が有する記憶部に加えて処理構造情報記憶部６２２を有し、サブルーチン情報記憶部１２２の変わりにサブルーチン情報記憶部６２１を有する。

【０２４７】

辞書情報合成部６１１は、辞書情報を参照してプログラム名や変数名などに対する補足情報を併記したサブルーチン情報を生成し、サブルーチン情報記憶部６２１に格納する処理部である。すなわち、サブルーチン情報記憶部６２１は、サブルーチン情報記憶部１２２が記憶する情報に加えて補足情報を記憶する。

【０２４８】

処理構造生成部６１２は、サブルーチン情報記憶部６２１に記憶されたサブルーチン情報を表形式に展開して処理構造情報記憶部６２２に格納し、利用者のオプション指定によって指定された情報を処理構造情報記憶部６２２に格納された表形式のサブルーチン情報に付加する処理部である。

【０２４９】

すなわち、処理構造情報記憶部６２２は、表形式に展開されたサブルーチン情報を記憶する記憶部である。また、オプション指定としては、サブルーチン呼出条件、引数、ジャンプ先、文の分類などの情報の処理構造図への付加を指定することができる。

【０２５０】

ドキュメント生成部６１３は、処理概要情報からプログラム仕様書を生成するとともに、処理構造情報記憶部６２２に記憶された表形式のサブルーチン情報から処理構造図を出力する。

【０２５１】

次に、辞書情報合成部６１１の処理手順について説明する。図５５は、辞書情報合成部６１１の処理手順を示すフローチャートである。同図に示すように、この辞書情報合成部６１１は、プログラム内のプログラム名、変数名、サブルーチン名などを取得し（ステップＳ６１１）、取得した情報を元に、辞書情報を参照して補足情報を検索する（ステップＳ６１２）。

【０２５２】

そして、補足情報が得られたか否かを判定し（ステップＳ６１３）、補足情報が得られた場合には、補足情報をサブルーチン情報記憶部６２１のサブルーチン情報に付加し、補

足情報が得られなかった場合には、そのまま処理を終了する（ステップＳ６１４）。

【０２５３】

このように、この辞書情報合成部６１１が辞書情報を参照し、辞書情報の参照によって得られた補足情報をサブルーチン情報記憶部６２１のサブルーチン情報に付加することによって、処理構造図に補足情報を加えることができる。

【０２５４】

次に、処理構造生成部６１２の処理手順について説明する。図５６は、処理構造生成部６１２の処理手順を示すフローチャートである。同図に示すように、この処理構造生成部６１２は、サブルーチン情報記憶部６２１に記憶されたサブルーチン情報を表形式に展開し、処理構造情報記憶部６２２に格納する（ステップＳ６２１）。

【０２５５】

そして、オプション指定を読み込み、利用者が条件文表示のオプションを指定しているか否かを調べ（ステップＳ６２２）、利用者が条件文表示のオプションを指定している場合には、サブルーチン呼出条件を検索し（ステップＳ６２３）、処理構造情報記憶部６２２のサブルーチン情報に付加する（ステップＳ６２４）。

【０２５６】

また、利用者が呼出プログラムの引数情報表示のオプションを指定しているか否かを調べ（ステップＳ６２５）、利用者が呼出プログラムの引数情報表示のオプションを指定している場合には、値を設定しているプログラム引数を取得し（ステップＳ６２６）、処理構造情報記憶部６２２のサブルーチン情報に引数情報を付加する（ステップＳ６２４）。

【０２５７】

また、利用者が呼出プログラムの制御文表示のオプションを指定しているか否かを調べ（ステップＳ６２８）、利用者が制御文表示のオプションを指定している場合には、ジャンプ文のジャンプ先の情報を取得し（ステップＳ６２９）、処理構造情報記憶部６２２のサブルーチン情報にジャンプ先の情報を付加する（ステップＳ６２ａ）。

【０２５８】

また、利用者がサブルーチン内の文の分類別表示のオプションを指定しているか否かを調べ（ステップＳ６２ｂ）、利用者がサブルーチン内の文の分類別表示のオプションを指定している場合には、サブルーチン内で文の数を「入力、出力、計算、代入、条件、呼出、その他」の７種類に分類して数を数え（ステップＳ６２ｃ）、処理構造情報記憶部６２２のサブルーチン情報に種類別の文の数を付加する（ステップＳ６２ｄ）。

【０２５９】

このように、この処理構造生成部６１２が、サブルーチン情報記憶部６２１に記憶されたサブルーチン情報を表形式に展開し、展開したサブルーチン情報にオプション指定に基づくオプション情報を付加することによって、オプション情報付の処理構造図を生成することができる。

【０２６０】

次に、本実施例６に係る仕様書生成装置６００が生成する処理構造図について説明する。図５７－１および図５７－２は、処理構造図を生成する対象プログラムを示す図である。図５７－１は呼出元のプログラムを示し、図５７－２は呼出先のプログラムを示す。

【０２６１】

図５７－１に示すように、呼出元のプログラムは、プログラム名がＰＲＯＧＡ（ＰＲＯＧＲＡＭ－ＩＤ　ＰＲＯＧＡ）であり、ＲＥＡＤ－ＳＥＣＴとＭＡＩＮ－ＳＥＣＴの二つのセクションを有する。また、ＭＡＩＮ－ＳＥＣＴは、ＰＲＯＧＷを呼び出し、図５７－２は、ＰＲＯＧＷを示している。

【０２６２】

図５８は、図５７－１および図５７－２に示したプログラムから生成される処理構造図の一例を示す図である。同図において、ＳＥＣＴＩＯＮ（１層目）～ＳＥＣＴＩＯＮ（３層目）の列は、プログラムの呼出構造を示し、読み取りファイルおよび書き込みファイルの列は、呼出構造に対応した入出力ファイルを示している。

【０２６３】
具体的には、この処理構造図は、ＰＲＯＧＡは２層目のセクションとしてＲＥＡＤ－ＳＥＣＴとＭＡＩＮ－ＳＥＣＴを有し、ＲＥＡＤ－ＳＥＣＴはＩＮＦＩＬＥを入力ファイルとして使用することを示している。さらに、ＭＡＩＮ－ＳＥＣＴはＣＡＬＬ文でＰＲＯＧＷを呼び出しており、ＰＲＯＧＷ内でＯＵＴＦＩＬＥを出力ファイルとして使用することを示している。
【０２６４】
上述してきたように、本実施例６では、辞書情報合成部６１１が辞書情報から補足情報を検索してサブルーチン情報に付加し、処理構造生成部６１２がサブルーチン情報を表形式に展開するとともに、展開したサブルーチン情報にオプション指定で指定された情報を付加し、ドキュメント生成部６１３が表形式のサブルーチン情報から補足情報やオプション情報付の処理構造図を生成することとしたので、プログラムの呼出構造と入出力情報との詳細な関係の把握を容易にすることができる。
【０２６５】
すなわち、処理構造図には、呼出先のサブルーチンを全て辿った入出力情報が表形式で示されているため、あるサブルーチンが下位のサブルーチンも含めてアクセスする可能性のある入出力情報の把握を容易にすることができる。
【実施例７】
【０２６６】
実施例６では、プログラムの処理構造の把握を容易にする処理構造図を生成する仕様書生成装置６００について説明したが、プログラムの実行時の動きを把握するためには、プログラムの静的な処理構造だけでなく、プログラムを実行して得られる実行ログ情報が有効となる。そこで、本実施例７では、プログラムの処理構造図に実行ログ情報を付加する仕様書生成装置について説明する。
【０２６７】
まず、本実施例７に係る仕様書生成装置の構成について説明する。図５９は、本実施例７に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図５４に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。
【０２６８】
同図に示すように、この仕様書生成装置７００は、制御部７１０と記憶部７２０とを有する。制御部７１０は、図５４に示した制御部６１０が有する機能部に加えて実行ログ情報解析部７１１と、実行情報合成部７１２とを有し、ドキュメント生成部６１３の代わりにドキュメント生成部７１３を有する。また、記憶部７２０は、図５４に示した記憶部６２０の処理構造情報記憶部６２２の代わりに処理構造情報記憶部７２１を有する。
【０２６９】
実行ログ情報解析部７１１は、プログラムを実行して得られる実行ログからサブルーチン単位での実行回数情報を抽出する処理部であり、抽出した実行回数を実行情報合成部７１２に渡す。
【０２７０】
実行情報合成部７１２は、実行ログ情報解析部７１１から受け取ったサブルーチン単位での実行回数に基づいて、処理構造図の各サブルーチンに対応する領域の色分けを行う処理部であり、色分けした情報を処理構造情報記憶部７２１に格納する。すなわち、処理構造情報記憶部７２１は、処理構造情報記憶部６２２が記憶する情報に加えて各サブルーチンの色情報を有する。
【０２７１】
ドキュメント生成部７１３は、処理概要情報からプログラム仕様書を生成するとともに、処理構造情報記憶部７２１に記憶された色分け情報付のサブルーチン情報から、色分け付の処理構造図を出力する処理部である。
【０２７２】

次に、実行ログ情報解析部７１１の処理手順について説明する。図６０は、実行ログ情報解析部７１１の処理手順を示すフローチャートである。同図に示すように、この実行ログ情報解析部７１１は、プログラムの実行ログがあるか否かを判定する（ステップＳ７１１）。

【０２７３】

そして、実行ログがある場合には、実行ログからサブルーチン単位の実行回数を取得し（ステップＳ７１２）、取得した実行回数をサブルーチン名とともに実行情報合成部７１２に渡す（ステップＳ７１３）。一方、実行ログがない場合には、そのまま処理を終了する。

【０２７４】

次に、実行情報合成部７１２の処理手順について説明する。図６１は、実行情報合成部７１２の処理手順を示すフローチャートである。同図に示すように、この実行情報合成部７１２は、実行ログ情報解析部７１１からサブルーチン名と実行回数を受け取り（ステップＳ７２１）、受け取ったサブルーチン名を用いて処理構造情報記憶部７２１を検索する（ステップＳ７２２）。

【０２７５】

そして、処理構造情報記憶部７２１からサブルーチンが検索できたか否かを調べ（ステップＳ７２３）、サブルーチンが検索できた場合には、そのサブルーチンの処理構造図における領域を実行回数に応じて色づけする情報を処理構造情報記憶部７２１に加える（ステップＳ７２４）。一方、サブルーチンが検索できなかった場合には、そのまま処理を終了する。

【０２７６】

このように、実行ログ情報解析部７１１が実行ログを解析してサブルーチンの実行回数を抽出し、実行情報合成部７１２がサブルーチンの実行回数に基づいて処理構造図におけるサブルーチンの領域を色分けすることによって、色分け付の処理構造図を生成することができる。

【０２７７】

次に、本実施例７に係る仕様書生成装置７００が生成する処理構造図について説明する。図６２は、図５７－１および図５７－２に示したプログラムから生成される処理構造図の一例を示す図である。

【０２７８】

同図に示すように、この処理構造図では、ＲＥＡＤ－ＳＥＣＴとＭＡＩＮ－ＳＥＣＴに対応する領域が実行回数に応じて異なる方向に網掛けされている。実際には、これらの領域に対して、網掛けでなく異なる色が割り付けられる。

【０２７９】

また、この処理構造図は、サブルーチン呼出条件、引数、ジャンプ先、文の分類などの情報を処理構造図に付加するオプション指定を利用者が行った場合を示しており、例えば、ＲＥＡＤ－ＳＥＣＴには、「入力：１、計算：１、代入：１」といった分類別ステートメント数が出力されている。

【０２８０】

上述してきたように、本実施例７では、実行ログ情報解析部７１１が実行ログを解析してサブルーチンの実行回数を抽出し、実行情報合成部７１２が実行ログ情報解析部７１１により抽出されたサブルーチン実行回数に基づいて色分けした情報を処理構造情報記憶部７２１に付加し、ドキュメント生成部７１３が実行情報合成部７１２により色分けられたサブルーチン情報を用いて色分け付の処理構造図を生成することとしたので、プログラムの実行時の動作の理解を容易にすることができる。

【０２８１】

なお、本実施例１～７の仕様書生成装置で生成されるプログラム仕様書などの仕様書では、仕様書内のデータまたはオブジェクトにハイパーリンクを設けることによって、仕様書間の相互参照、または他のドキュメントなどへの参照を可能にすることもできる。

【０２８２】
例えば、画面遷移図の場合、図中のプログラム名をクリックするとプログラム概要書へジャンプしたり、画面図形をクリックすると他のツールで生成した画面レイアウト図へジャンプしたりすることを可能とすることができる。
【０２８３】
また、本実施例１～７では、仕様書生成装置について説明したが、この仕様書生成装置が有する構成をソフトウェアによって実現することで、同様の機能を有する仕様書生成プログラムを得ることができる。そこで、この仕様書生成プログラムを実行するコンピュータシステムについて説明する。
【０２８４】
図６３は、本実施例１～７に係る仕様書生成プログラムを実行するコンピュータシステムを示す図である。同図に示すように、このコンピュータシステム８００は、本体部８０１と、本体部８０１からの指示により表示画面８０２ａに情報を表示するディスプレイ８０２と、このコンピュータシステム８００に種々の情報を入力するためのキーボード８０３と、ディスプレイ８０２の表示画面８０２ａ上の任意の位置を指定するマウス８０４と、ＬＡＮ８０６または広域エリアネットワーク（ＷＡＮ）に接続するＬＡＮインタフェースと、公衆回線８０７に接続するモデムとを有する。ここで、ＬＡＮ８０６は、他のコンピュータシステム（ＰＣ）８１１、サーバ８１２、プリンタ８１３などとコンピュータシステム８００とを接続している。
【０２８５】
また、図６４は、図６３に示した本体部８０１の構成を示す機能ブロック図である。同図に示すように、この本体部８０１は、ＣＰＵ８２１と、ＲＡＭ８２２と、ＲＯＭ８２３と、ハードディスクドライブ（ＨＤＤ）８２４と、ＣＤ－ＲＯＭドライブ８２５と、ＦＤドライブ８２６と、Ｉ／Ｏインタフェース８２７と、ＬＡＮインタフェース８２８と、モデム８２９とを有する。
【０２８６】
そして、このコンピュータシステム８００において実行される仕様書生成プログラムは、フロッピィディスク（ＦＤ）８０８、ＣＤ－ＲＯＭ８０９、ＤＶＤディスク、光磁気ディスク、ＩＣカードなどの可搬型記憶媒体に記憶され、これらの記憶媒体から読み出されてコンピュータシステム８００にインストールされる。
【０２８７】
あるいは、この仕様書生成プログラムは、ＬＡＮインタフェース８２８を介して接続されたサーバ８１２のデータベース、他のコンピュータシステム（ＰＣ）８１１のデータベースなどに記憶され、これらのデータベースから読み出されてコンピュータシステム８００にインストールされる。
【０２８８】
そして、インストールされた仕様書生成プログラムは、ＨＤＤ８２４に記憶され、ＲＡＭ８２２、ＲＯＭ８２３などを利用してＣＰＵ８２１により実行される。
【０２８９】
また、本実施例では、ＣＯＢＯＬで開発されたプログラムから仕様書を生成する場合について説明したが、本発明はこれに限定されるものではなく、ＦＯＲＴＲＡＮやＣなどの他のプログラミング言語で開発されたプログラムから仕様書を生成する場合にも同様に適用することができる。
【０２９０】
（付記１）ソースプログラムを解析して仕様書を生成する仕様書生成プログラムであって、
前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、
前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、

前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と
をコンピュータに実行させることを特徴とする仕様書生成プログラム。
【０２９１】
（付記２）前記処理概要情報生成手順は、特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを入出力付構造情報から抽出して前記処理概要情報を生成し、
前記文書生成手順は、前記特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを用いてプログラム仕様書を生成することを特徴とする付記１に記載の仕様書生成プログラム。
【０２９２】
（付記３）前記文書生成手順は、サブルーチンの呼出階層を列とし、各サブルーチンの名前をサブルーチンの呼出階層値に対応する列に表示する表形式の呼出構造図をプログラム仕様書に含めることを特徴とする付記２に記載の仕様書生成プログラム。
【０２９３】
（付記４）前記入出力付構造情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、
前記処理概要情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、
前記文書生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成することを特徴とする付記１に記載の仕様書生成プログラム。
【０２９４】
（付記５）前記文書生成手順は、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出するコメント抽出手順と、
新たに生成するプログラム仕様書の所定の位置に、前記コメント抽出手順により抽出されたコメントを継承するコメント継承手順とを
さらにコンピュータに実行させることを特徴とする付記１、２または４に記載の仕様書生成プログラム。
【０２９５】
（付記６）前記ソースプログラムを構成する文をまとめることにより該ソースプログラムのプログラム概要情報を生成するプログラム概要情報生成手順と、
前記プログラム概要情報生成手順により生成されたプログラム概要情報から自然言語によるプログラム概要文を生成するプログラム概要文生成手順と、
前記プログラム概要文生成手順により生成されたプログラム概要文を用いて前記ソースプログラムのプログラム概要書を生成するプログラム概要書生成手順と
をさらにコンピュータに実行させることを特徴とする付記１、２または４に記載の仕様書生成プログラム。
【０２９６】
（付記７）前記プログラム概要情報生成手順は、前記ソースプログラムに含まれるデータの重要度を決定するデータ重要度決定手順と、
前記ソースプログラムを構成する文の重要度を前記データ重要度決定手順により決定されたデータの重要度を用いて決定する文重要度決定手順と、
前記ソースプログラムを構成する文を前記文重要度決定手順により決定された文の重要度を用いてまとめることによって前記プログラム概要情報を生成するまとめ処理手順と
をコンピュータに実行させることを特徴とする付記６に記載の仕様書生成プログラム。
【０２９７】
（付記８）バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出するステップ入出力情報抽出手順と、
前記ステップ入出力情報抽出手順により抽出されたジョブステップの入出力情報に基づいて前記バッチジョブ全体の入力情報および出力情報を特定するジョブ入出力情報特定手

順と、
　前記ジョブ入出力情報特定手順により特定された入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出するプログラム情報抽出手順と、
　前記ジョブ入出力情報特定手順により特定された入力情報および出力情報、ならびに前記プログラム情報抽出手順により抽出されたプログラムの情報を用いて前記バッチジョブのバッチジョブ処理概要書を生成するバッチジョブ処理概要書生成手順と
　をさらにコンピュータに実行させることを特徴とする付記１、２または４に記載の仕様書生成プログラム。
【０２９８】
（付記９）画面についての情報を定義した画面定義体を解析して画面遷移情報を作成する画面遷移情報作成手順と、
　前記画面遷移情報作成手順により作成された画面遷移情報を用いて画面遷移図を生成する画面遷移図生成手順と
　をさらにコンピュータに実行させることを特徴とする付記１、２または４に記載の仕様書生成プログラム。
【０２９９】
（付記１０）前記画面定義体は、画面とプログラムとの間の遷移に関する情報を含み、
　前記画面遷移情報作成手順により作成される画面遷移情報は、画面とプログラムとの間の遷移を含み、
　画面とプログラムとの間の遷移を含む画面遷移情報と前記入出力付構造情報生成手順により生成された入出力付構造情報とをマージし、画面とプログラムとの遷移および該プログラムの入出力をマージしたマージ図面情報を作成するマージ手順をさらにコンピュータに実行させ、
　前記画面遷移図生成手順は、前記マージ手順により作成されたマージ図面情報を用いて画面とプログラムとの遷移および該プログラムの入出力をマージした図面を生成することを特徴とする付記９に記載の仕様書生成プログラム。
【０３００】
（付記１１）ソースプログラムを解析して仕様書を生成する仕様書生成プログラムを記録したコンピュータ読み取り可能な記録媒体であって、
　前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、
　前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、
　前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と
　をコンピュータに実行させる仕様書生成プログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。
【０３０１】
（付記１２）ソースプログラムを解析して仕様書を生成する仕様書生成装置であって、
　前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手段と、
　前記入出力付構造情報生成手段により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手段と、
　前記処理概要情報生成手段により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手段と
　を備えたことを特徴とする仕様書生成装置。
【０３０２】
（付記１３）ソースプログラムを解析して仕様書を生成する仕様書生成方法であって、
　前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入

出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成工程と、
前記入出力付構造情報生成工程により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成工程と、
前記処理概要情報生成工程により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成工程と
を含んだことを特徴とする仕様書生成方法。

【産業上の利用可能性】

【０３０３】

以上のように、本発明にかかる仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法は、ソフトウェアの保守に有用であり、特に、大規模なプログラムの処理の概要を把握したい場合に適している。

【図面の簡単な説明】

【０３０４】

【図１】本実施例１に係る仕様書生成装置の構成を示す機能ブロック図である。
【図２】構造抽出部が抽出するサブルーチン呼出の構造を説明するための説明図である。
【図３】サブルーチン情報記憶部が記憶するサブルーチン情報のデータ構造の一例を示す図である。
【図４】サブルーチン情報のリストのつながり（ツリー構造）を説明するための説明図である。
【図５】図３に示した呼出条件リストのデータ構造の一例を示す図である。
【図６】図３に示した入出力情報リストのデータ構造の一例を示す図である。
【図７】処理概要生成部が生成する処理概要情報を説明するための説明図である。
【図８】ドキュメント生成部が生成するプログラム仕様書の一例を示す図である。
【図９】構造抽出部によるサブルーチン情報生成処理および条件分岐情報抽出部による条件分岐情報抽出処理の処理手順を示すフローチャートである。
【図１０】入出力データ抽出部による入出力情報抽出処理の処理手順を示すフローチャートである。
【図１１】構造－入出力データ関連生成部による入出力情報のサブルーチン情報記憶部への追加処理の処理手順を示すフローチャートである。
【図１２】処理概要生成部による処理概要生成処理の処理手順を示すフローチャートである。
【図１３】本実施例２に係るコメント継承を説明するための説明図である。
【図１４】本実施例２に係る仕様書生成装置の構成を示す機能ブロック図である。
【図１５】コメント記述領域リストに対応するプロパティのデータ構造の一例を示す図である。
【図１６－１】コメント記述領域に対応するプロパティのデータ構造の一例を示す図である。
【図１６－２】複数の記述領域がある場合のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。
【図１７】図１４に示したコメント記入内容継承部の処理手順を示すフローチャートである。
【図１８】スプレッドシートを用いたプログラム仕様書のコメント記述領域の一例を示す図である。
【図１９】スプレッドシートを用いたプログラム仕様書のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。
【図２０】スプレッドシートを用いたプログラム仕様書のコメント記述領域（複数の記述領域がある場合）に対応するプロパティのデータ構造の一例を示す図である。
【図２１】処理構造図におけるコメント記述領域を示す図である。
【図２２】図２１に示した処理構造図における各コメント記述領域の位置情報データ

の一例を示す図である。
【図２３】本実施例３に係る仕様書生成装置の構成を示す機能ブロック図である。
【図２４】プログラム概要生成部の構成を示す機能ブロック図である。
【図２５－１】プログラムソース例を示す図である。
【図２５－２】図２５－１に示したプログラムソース例から生成されるプログラム中間情報を示す図である。
【図２６】データの重要度の分類の一例を示す図である。
【図２７】プログラム中間情報の変数用タグに付加される重要度の一例を示す図である。
【図２８】ステートメントの重要度の分類の一例を示す図である。
【図２９】プログラム中間情報のステートメントのタグに付加される重要度の一例を示す図である。
【図３０】まとめ管理部の初期動作を説明するための説明図である。
【図３１】まとめ管理部とまとめ処理プログラムとのインタフェース（Ｉ／Ｆ）を示す図である。
【図３２】まとめ処理プログラムの実行順序保持の一例を示す図である。
【図３３－１】まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その１）である。
【図３３－２】まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その２）である。
【図３３－３】まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その３）である。
【図３３－４】まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その４）である。
【図３３－５】まとめ処理プログラムが行うまとめのルール（全体的）の例を示す図（その１）である。
【図３３－６】まとめ処理プログラムが行うまとめのルール（全体的）の例を示す図（その２）である。
【図３４】日本語変換部の処理を説明するための説明図である。
【図３５－１】その他の日本語テンプレートの例を示す図（その１）である。
【図３５－２】その他の日本語テンプレートの例を示す図（その２）である。
【図３６】プログラムサンプルを示す図である。
【図３７】まとめ管理部が利用するまとめ処理プログラムのリストを示す図である。
【図３８】まとめ処理プログラムの実行順序を示す図である。
【図３９】まとめ処理終了時のプログラム中間情報を示す図である。
【図４０】日本語変換部が生成した日本語情報を示す図である。
【図４１】ドキュメント生成部が生成するプログラム概要書の一例を示す図である。
【図４２】本実施例４に係る仕様書生成装置の構成を示す機能ブロック図である。
【図４３】ジョブステップ入出力データ抽出部による入出力判定処理の処理手順を示すフローチャートである。
【図４４】全体入出力情報抽出部によるジョブ全体としてのファイル入出力判定処理の処理手順を示すフローチャートである。
【図４５】ユーティリティ除外処理の処理手順を示すフローチャートである。
【図４６】バッチジョブ記述の一例を示す図である。
【図４７】図４６に示したバッチジョブ記述からバッチジョブ記述言語構文解析部が生成するバッチジョブ解析情報を示す図である。
【図４８】図４６に示したバッチジョブ記述から生成されるバッチジョブ全体の処理概要を示す図である。
【図４９】本実施例５に係る仕様書生成装置の構成を示す機能ブロック図である。
【図５０】画面定義体の一例を示す図である。

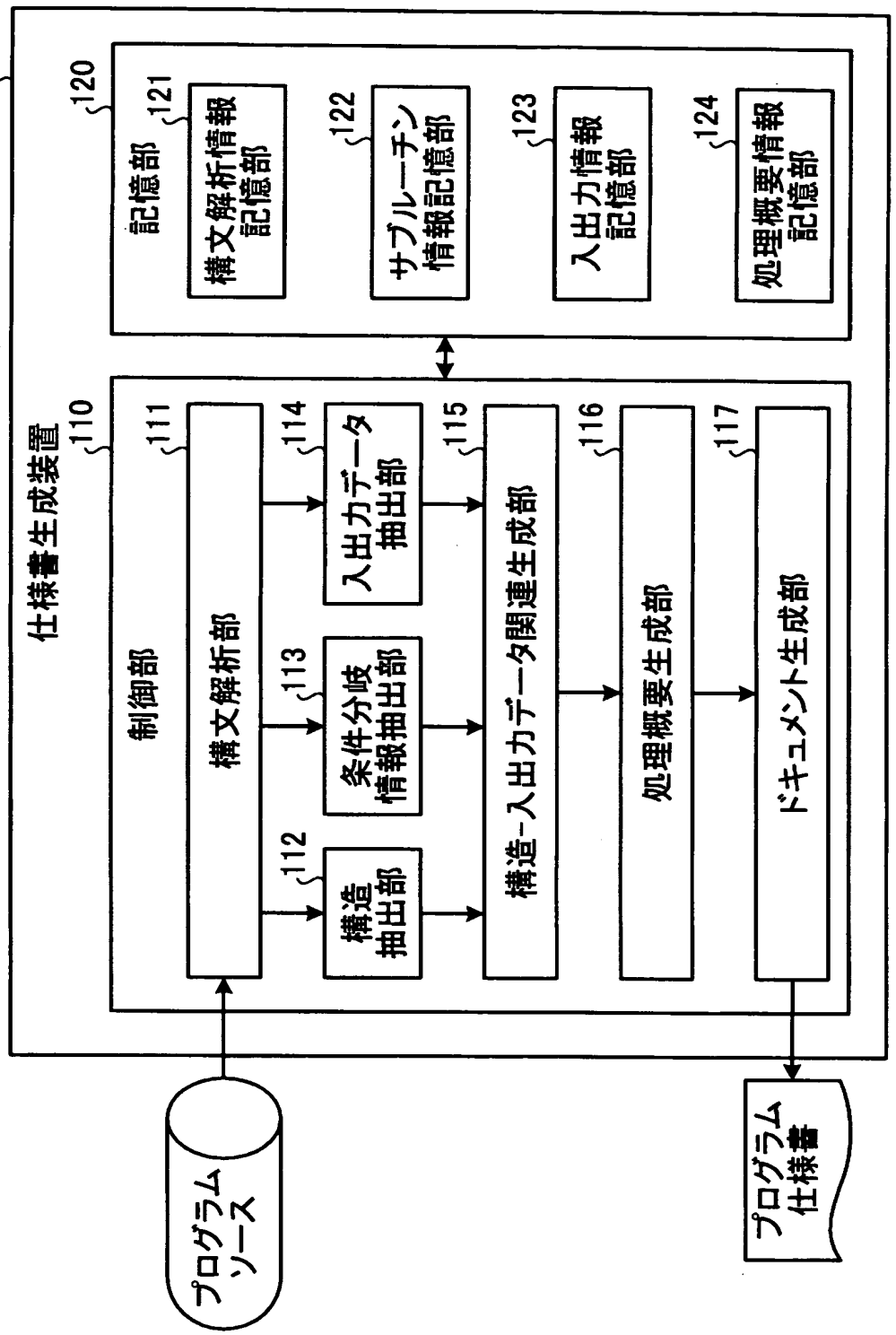
【図５１】画面遷移情報の一例を示す図である。
【図５２】画面遷移データの一例を示す図である。
【図５３－１】画面遷移図の一例を示す図である。
【図５３－２】画面遷移図の他の例を示す図である。
【図５４】本実施例６に係る仕様書生成装置の構成を示す機能ブロック図である。
【図５５】辞書情報合成部の処理手順を示すフローチャートである。
【図５６】処理構造生成部の処理手順を示すフローチャートである。
【図５７－１】処理構造図を生成する対象プログラム（呼出元）を示す図である。
【図５７－２】処理構造図を生成する対象プログラム（呼出先）を示す図である。
【図５８】図５７－１および図５７－２に示したプログラムから生成される処理構造図の一例を示す図である。
【図５９】本実施例７に係る仕様書生成装置の構成を示す機能ブロック図である。
【図６０】実行ログ情報解析部の処理手順を示すフローチャートである。
【図６１】実行情報合成部の処理手順を示すフローチャートである。
【図６２】図５７－１および図５７－２に示したプログラムから生成される処理構造図の一例を示す図である。
【図６３】本実施例１～７に係る仕様書生成プログラムを実行するコンピュータシステムを示す図である。
【図６４】図６３に示した本体部の構成を示す機能ブロック図である。
【図６５】利用者記入欄を設けないコメント継承の一例を示す図である。

【符号の説明】
【０３０５】
１００，２００，３００，４００，５００，６００，７００　仕様書生成装置
１１０，２１０，３１０，４１０，５１０，６１０，７１０　制御部
１１１　構文解析部
１１２　構造抽出部
１１３　条件分岐情報抽出部
１１４　入出力データ抽出部
１１５　構造－入出力データ関連性生成部
１１６，４１５　処理概要生成部
１１７，２１１，３１２，４１６，５１３，６１３，７１３　ドキュメント生成部
１２０，３２０，５２０，６２０，７２０　記憶部
１２１　構文解析情報記憶部
１２２，６２１　サブルーチン情報記憶部
１２３　入出力情報記憶部
１２４　処理概要情報記憶部
２１１ａ　コメント記入内容継承部
３１１　プログラム概要生成部
３１１ａ　データ重要度設定部
３１１ｂ　ステートメント重要度設定部
３１１ｃ　まとめ管理部
３１１ｄ　日本語変換部
３２１　まとめ処理記憶部
３２２　日本語テンプレート記憶部
３２３　プログラム概要情報記憶部
４１１　バッチジョブ記述言語構文解析部
４１２　ジョブステップ入出力データ抽出部
４１３　全体入出力情報抽出部
４１４　主要プログラム判定部
５１１　画面定義体解析部

５１２　画面遷移データ生成部
５２１　画面遷移情報記憶部
５２２　レイアウト情報記憶部
５２３　画面遷移データ記憶部
６１１　辞書情報合成部
６１２　処理構造生成部
６２２，７２１　処理構造情報記憶部
７１１　実行ログ情報解析部
７１２　実行情報合成部
８００，８１１　コンピュータシステム
８０１　本体部
８０２　ディスプレイ
８０２ａ　表示画面
８０３　キーボード
８０４　マウス
８０６　ＬＡＮ
８０７　公衆回線
８０８　フロッピィディスク
８０９　ＣＤ－ＲＯＭ
８１２　サーバ
８１３　プリンタ
８２１　ＣＰＵ
８２２　ＲＡＭ
８２３　ＲＯＭ
８２４　ハードディスクドライブ
８２５　ＣＤ－ＲＯＭドライブ
８２６　フロッピィディスクドライブ
８２７　Ｉ／Ｏインタフェース
８２８　ＬＡＮインタフェース
８２９　モデム

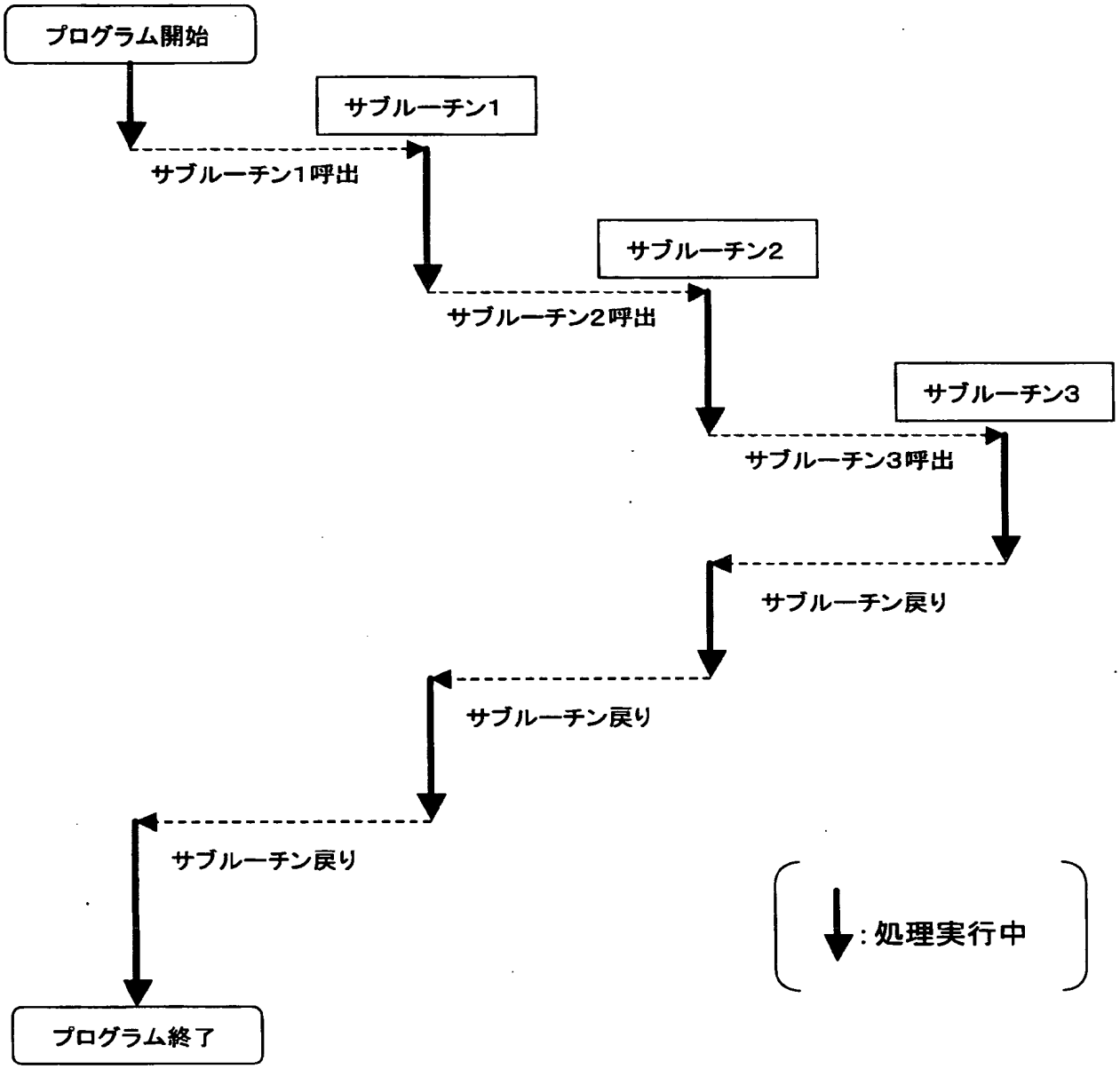
【書類名】図面
【図１】

本実施例1に係る仕様書生成装置の構成を示す機能ブロック図



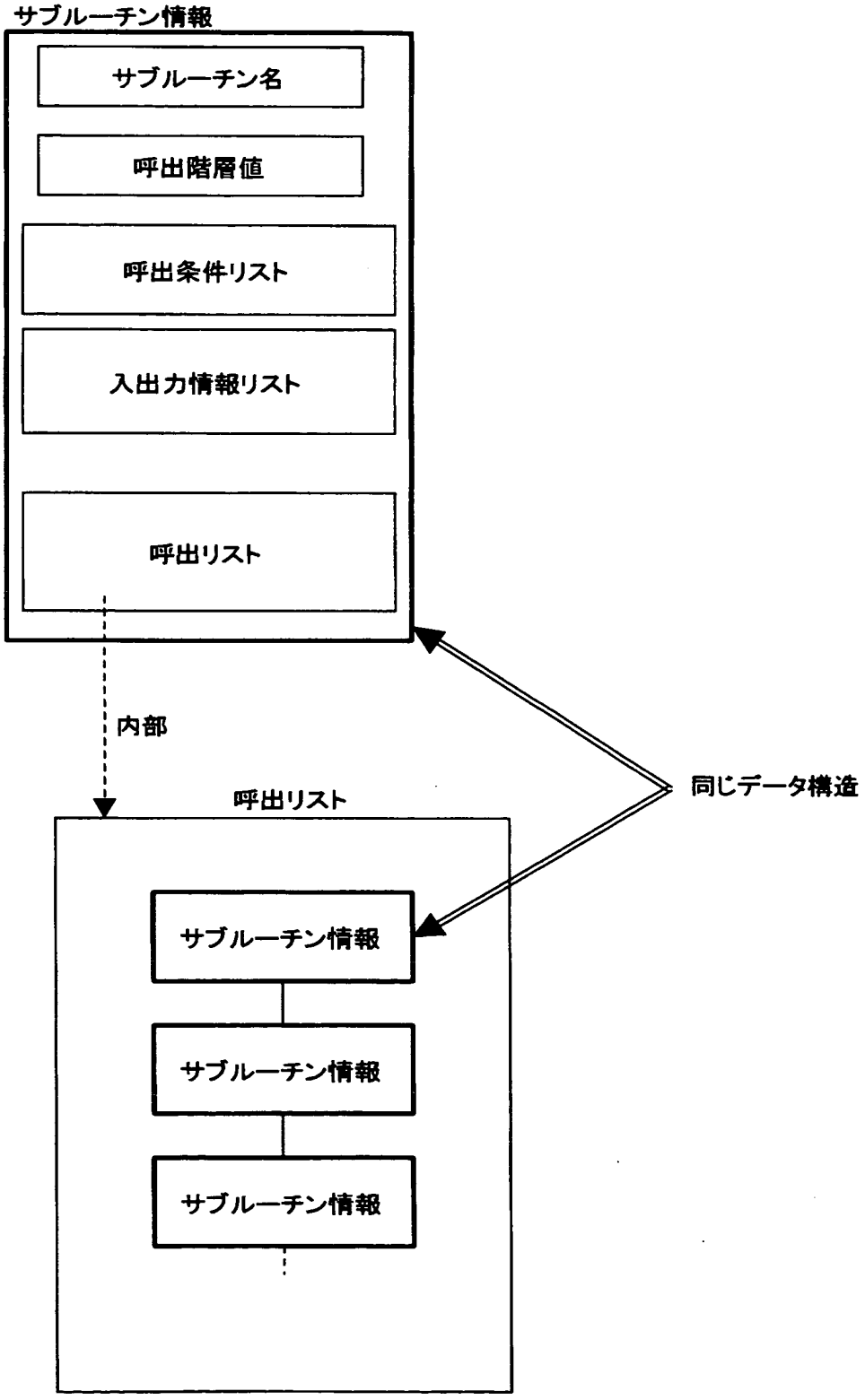
【図２】

構造抽出部が抽出するサブルーチン呼出の構造を
説明するための説明図



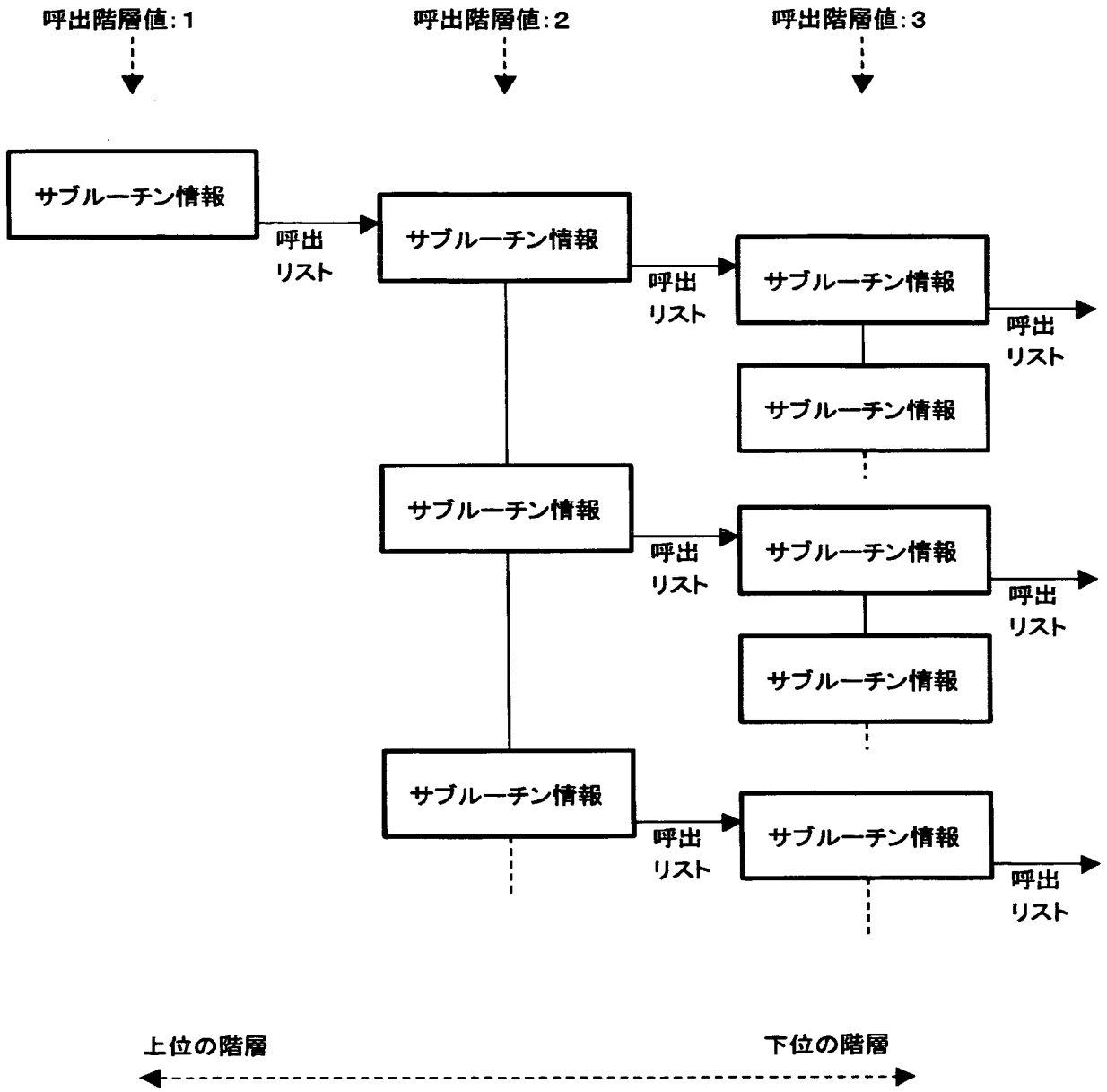
【図３】

サブルーチン情報記憶部が記憶するサブルーチン情報の
データ構造の一例を示す図



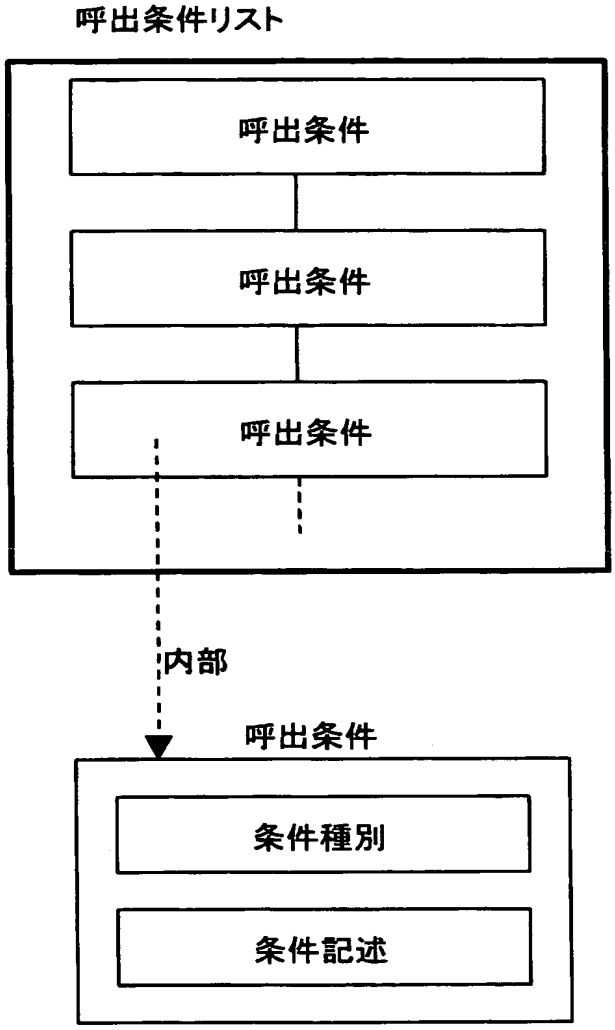
【図４】

サブルーチン情報のリストのつながり(ツリー構造)を
説明するための説明図



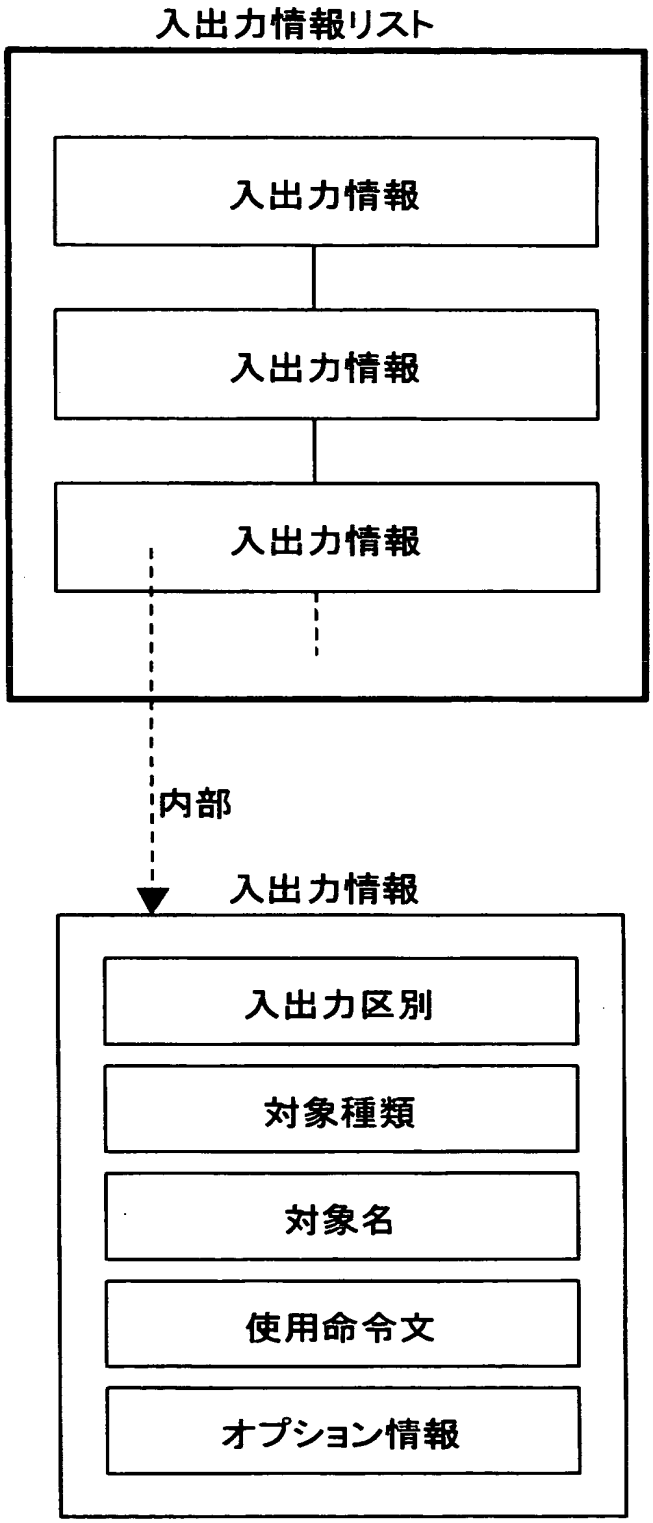
【図５】

図3に示した呼出条件リストのデータ構造の一例を示す図



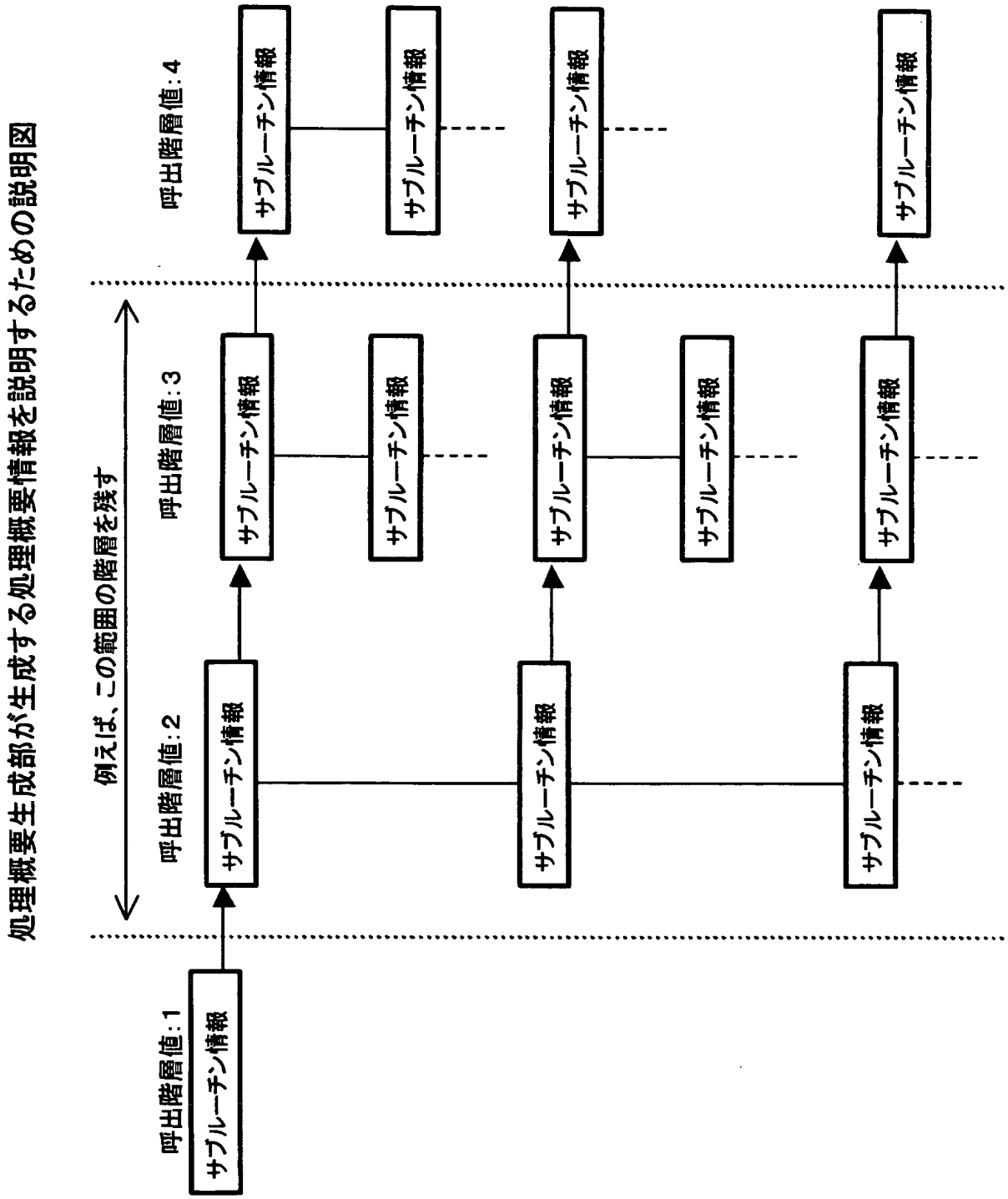
【図６】

図3に示した入出力情報リストのデータ構造の一例を示す図



【図７】

処理概要生成部が生成する処理概要情報を説明するための説明図



【図８】

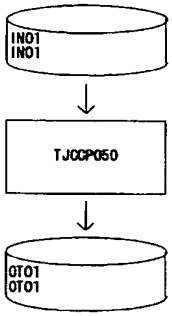
ドキュメント生成部が生成するプログラム仕様書の一例を示す図

プログラム名	TJCCP050	ファイル名	TJCCP050.scob

［コメント欄］

業務内容などを記載。

［入出力関連図］



［共通領域］

No.	レコード名	詳細
1	SQLSTATE	
2	SQLMSG	

［ファイル情報］

No.	ファイル名	外部名	種類	編成	登録集名	詳細
1	IN01	IN01	COBOL ファイル	行順	TJCCF031.cbl	
2	OT01	OT01	COBOL ファイル	行順	TJCCF051.cbl	

［入出力とセクション名］

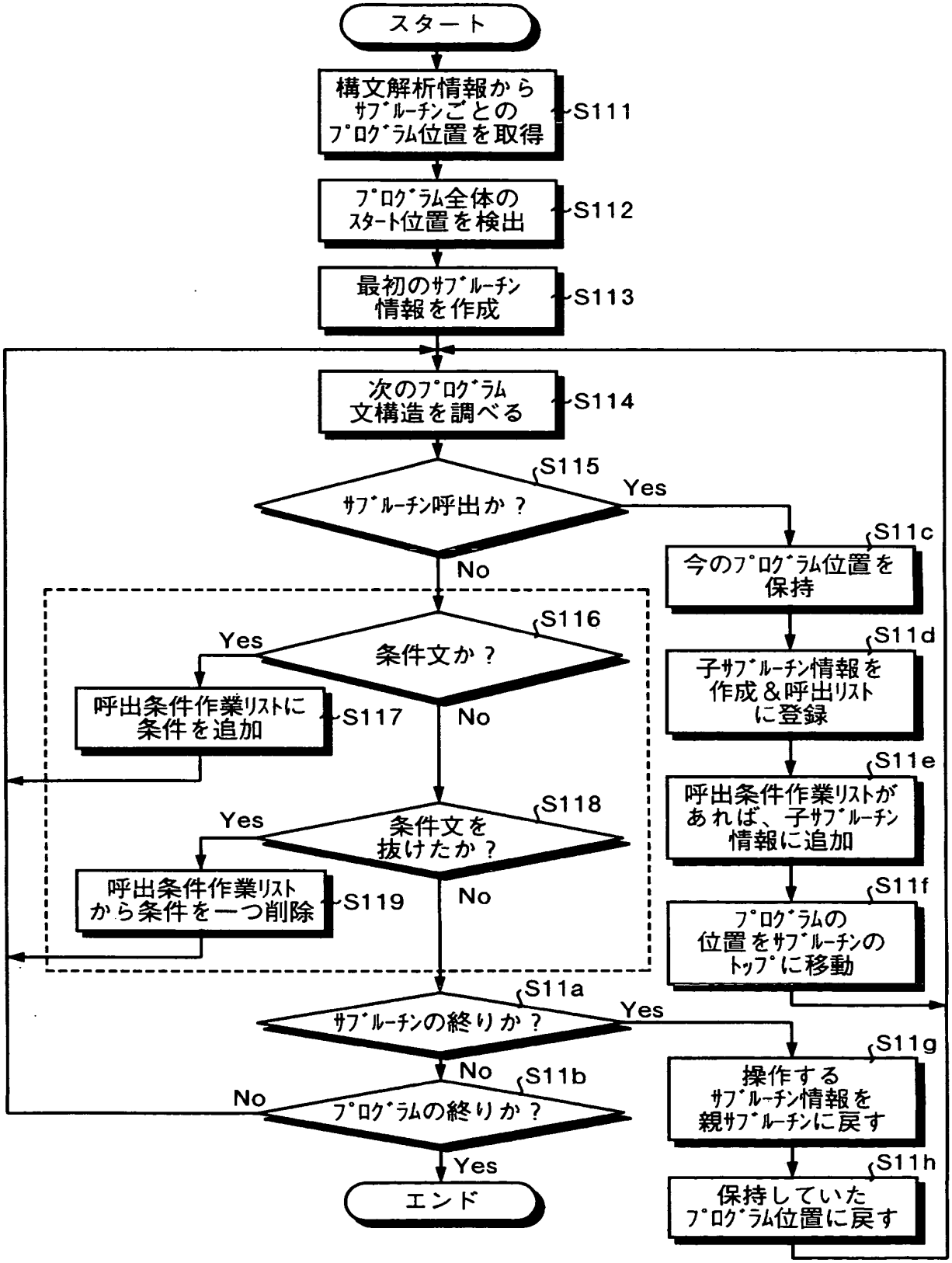
［入力］
TJSC.申込ファイル、TJSC.入居ファイル、TJSC.住戸マスタ、TJSC.使用料区分マスタ、TJSC.調定額確定ファイル、TJSC.名義ファイル、TJSC.コードマスタ、SC_B101001.MB101012、SC_B101001.MB101014、IN01
［処理］
パラメタチェック処理、読込処理、詳細処理（データチェック処理、調定額確定Ｆ読込処理、名義Ｆ読込処理、編集処理）
［出力］
TJCCF051

［処理構造図］（セクションの 2から3 層目を表示）

←呼び出し元	［セクション呼び出し構造］	呼び出し先→
パラメタチェック処理		
読込処理		
詳細処理	データチェック処理	

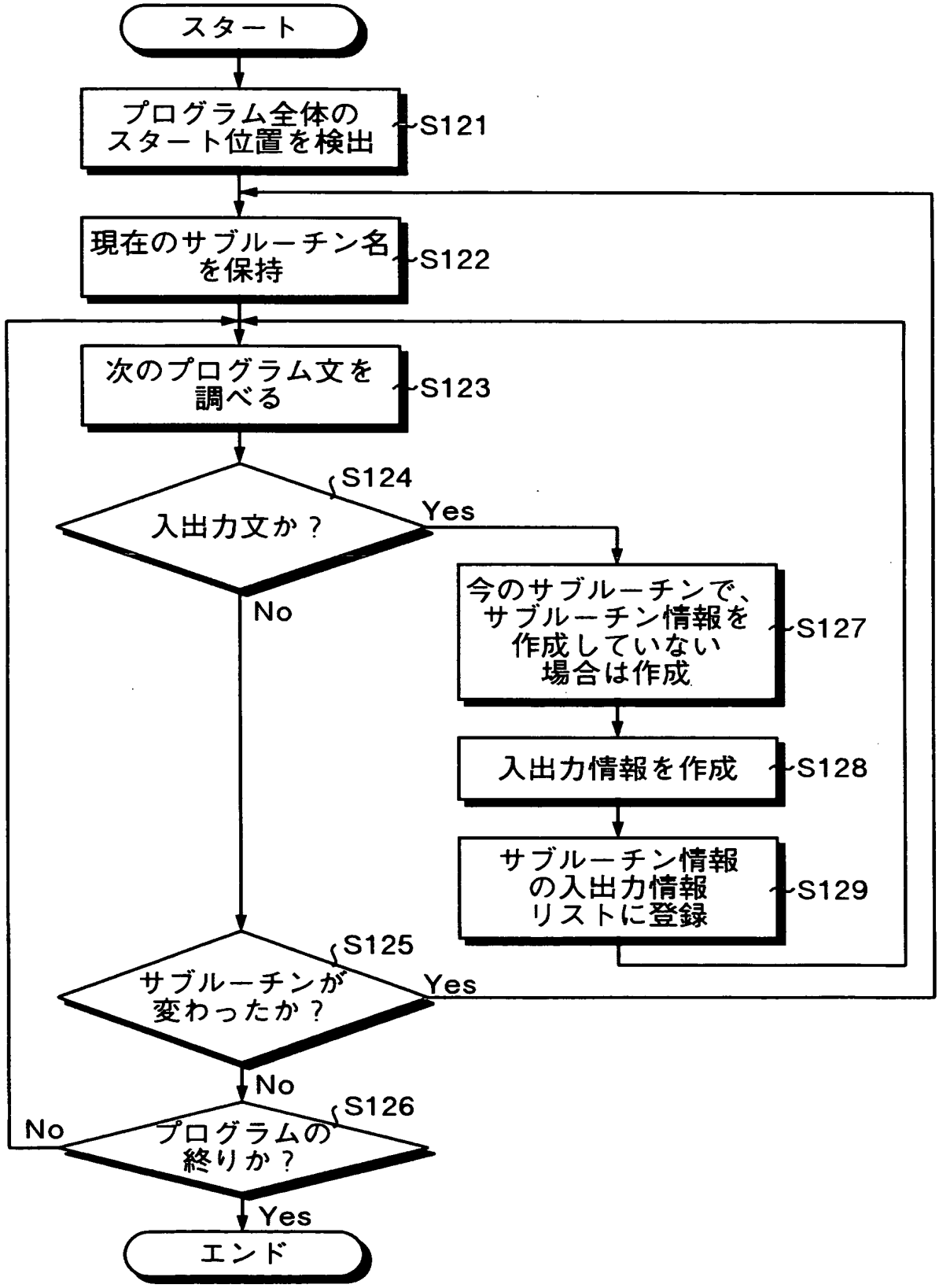
【図９】

構造抽出部によるサブルーチン情報生成処理および条件分岐
情報抽出部による条件分岐情報抽出処理の処理手順を示すフローチャート



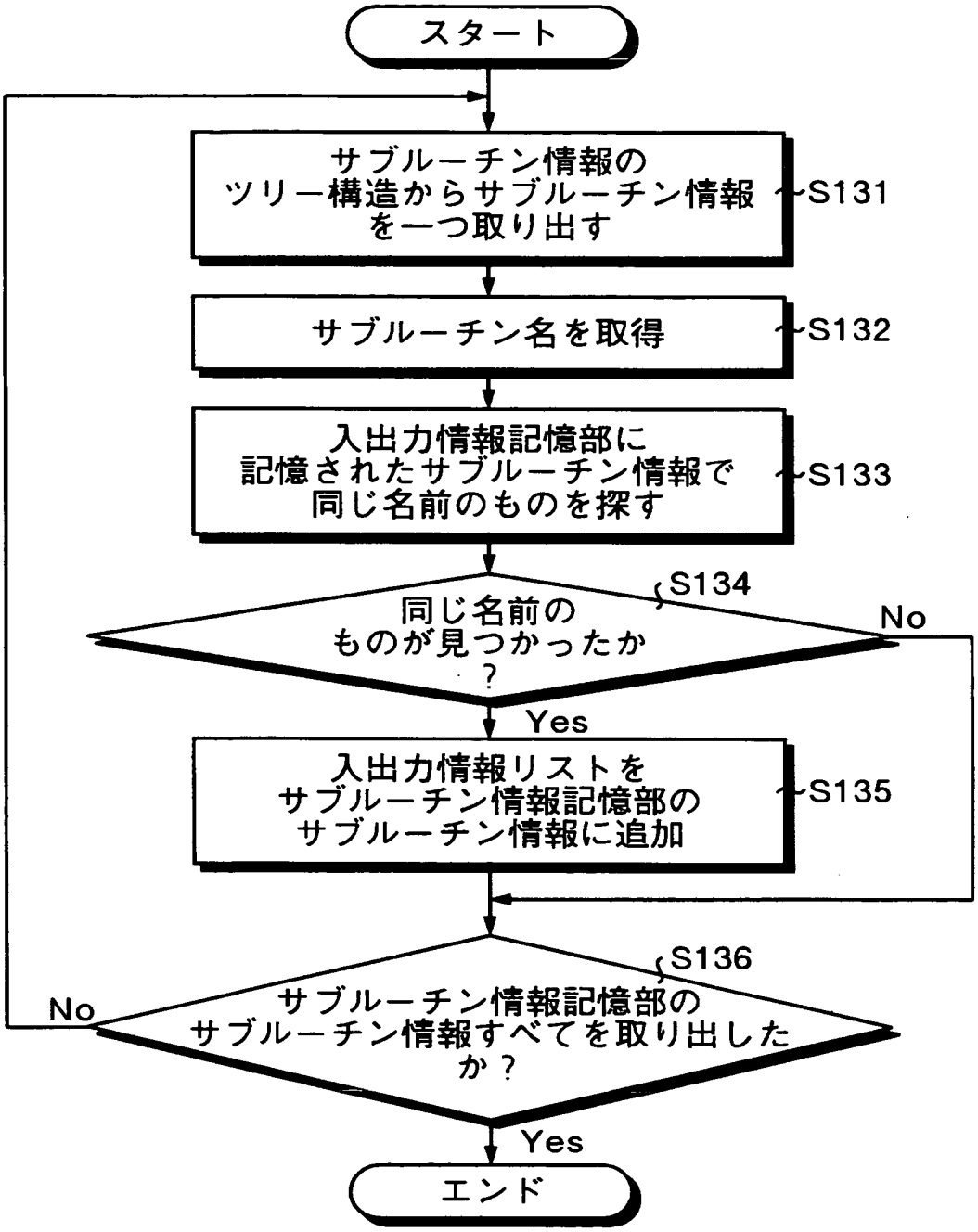
【図１０】

入出力データ抽出部による入出力情報抽出処理の
処理手順を示すフローチャート



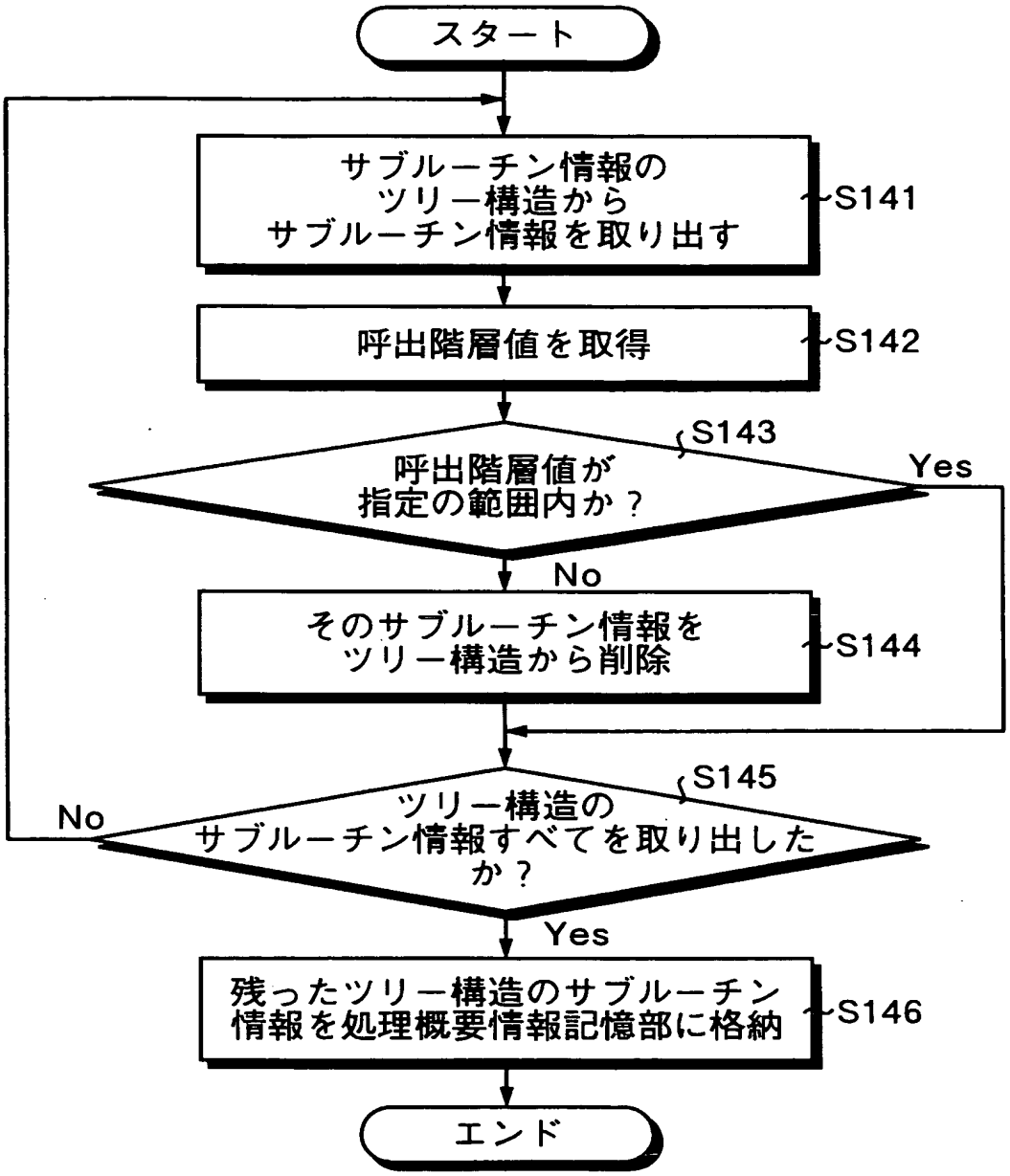
【図１１】

構造－入出力データ関連生成部による入出力情報のサブルーチン情報記憶部への追加処理の処理手順を示すフローチャート

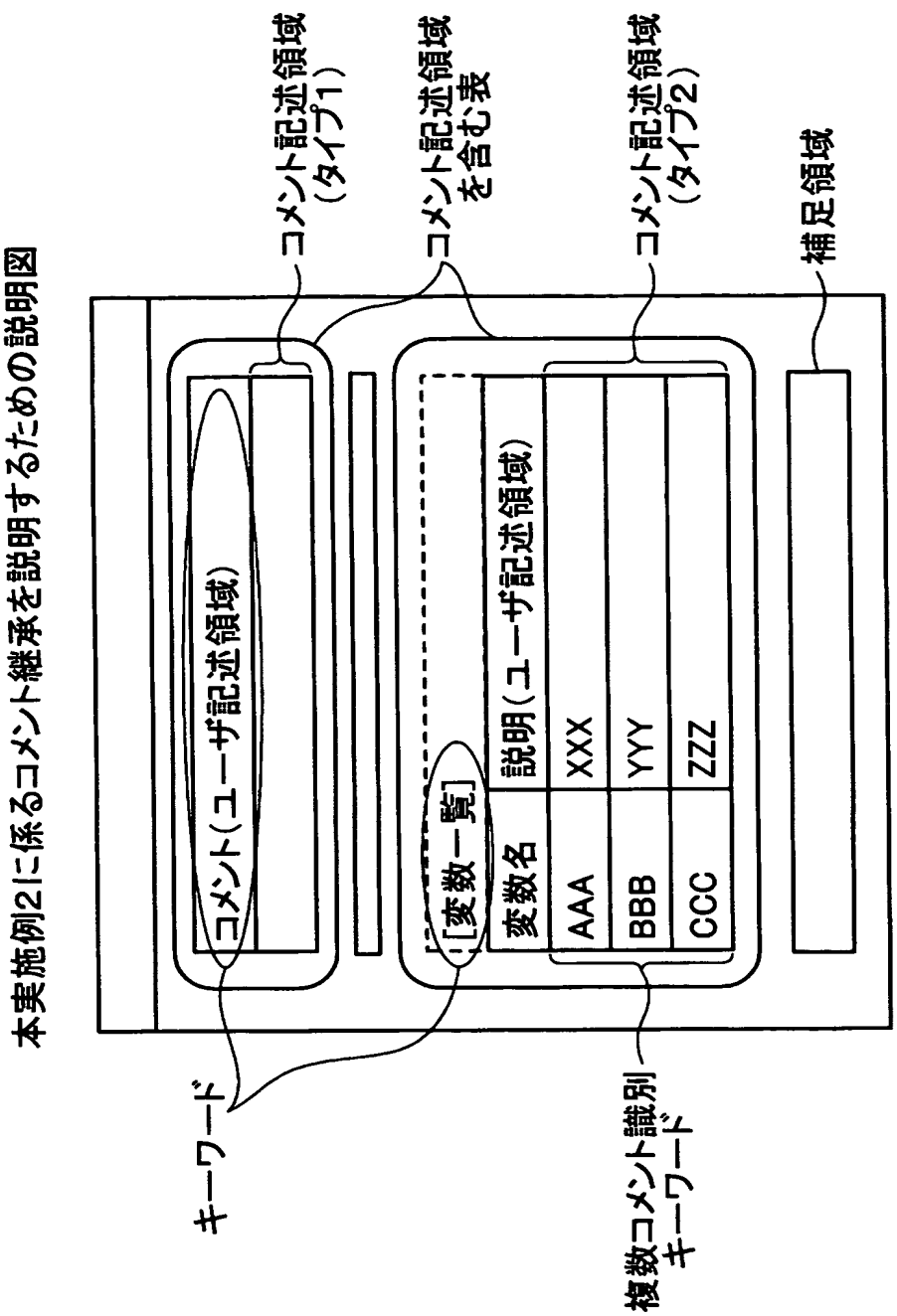


【図１２】

処理概要生成部による処理概要生成処理の処理手順を示すフローチャート

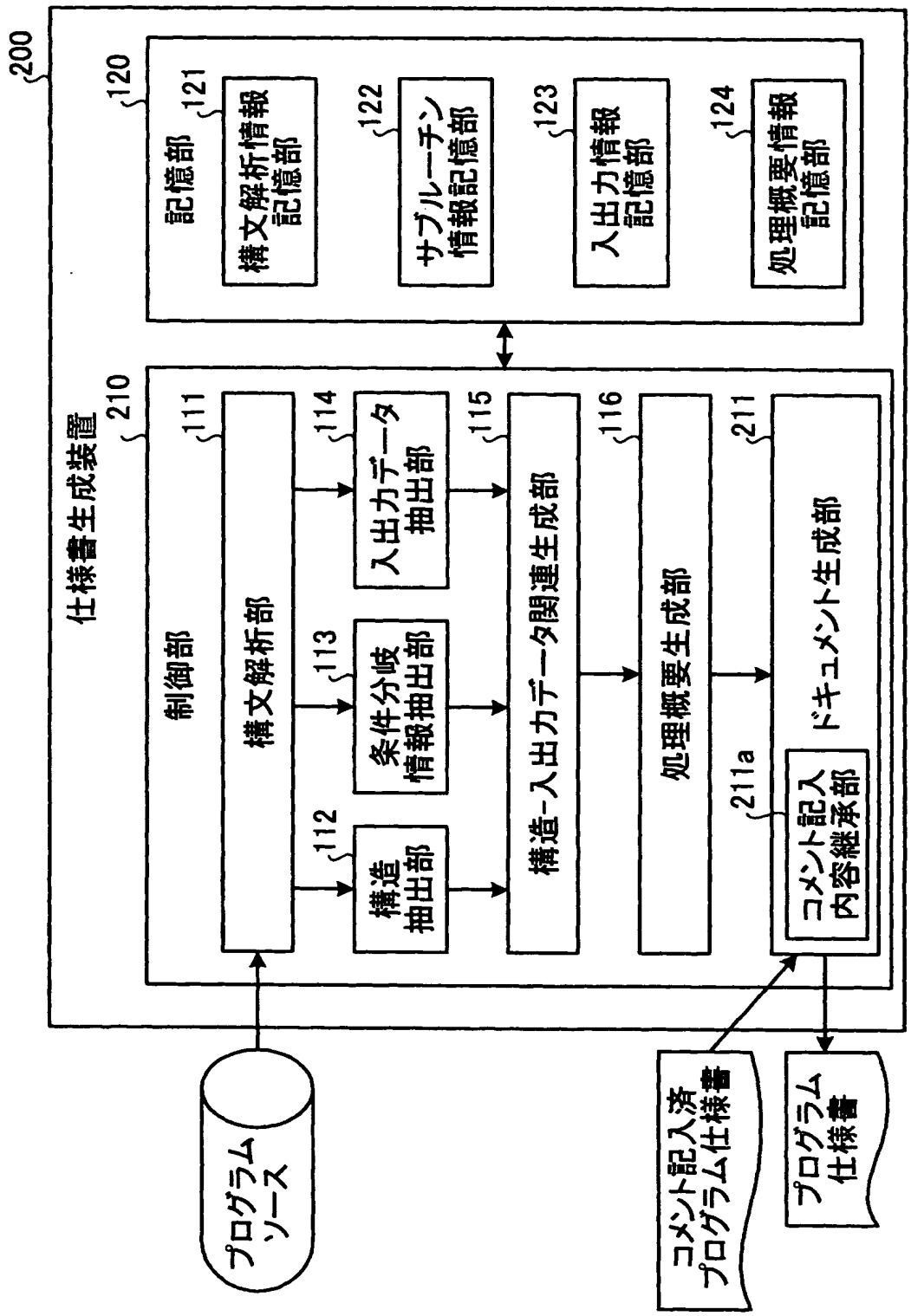


【図１３】


本実施例2に係るコメント継承を説明するための説明図
コメント記述領域（タイプ1）
コメント記述領域を含む表
コメント記述領域（タイプ2）
補足領域
コメント（ユーザ記述領域）
[変数一覧]
変数名
説明（ユーザ記述領域）
AAA
BBB
CCC
XXX
YYY
ZZZ
キーワード
複数コメント識別キーワード

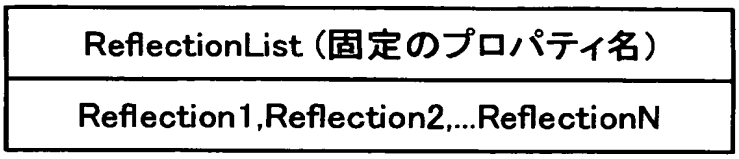
【図１４】

本実施例2に係る仕様書生成装置の構成を示す機能ブロック図



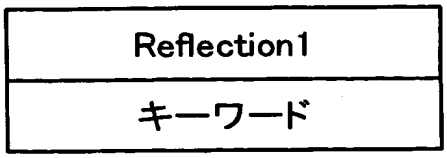
【図１５】

コメント記述領域リストに対応するプロパティのデータ構造の一例を示す図



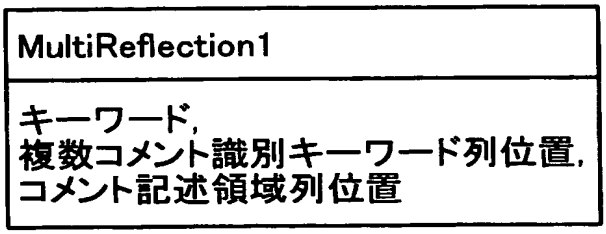
【図１６－１】

コメント記述領域に対応するプロパティのデータ構造の一例を示す図



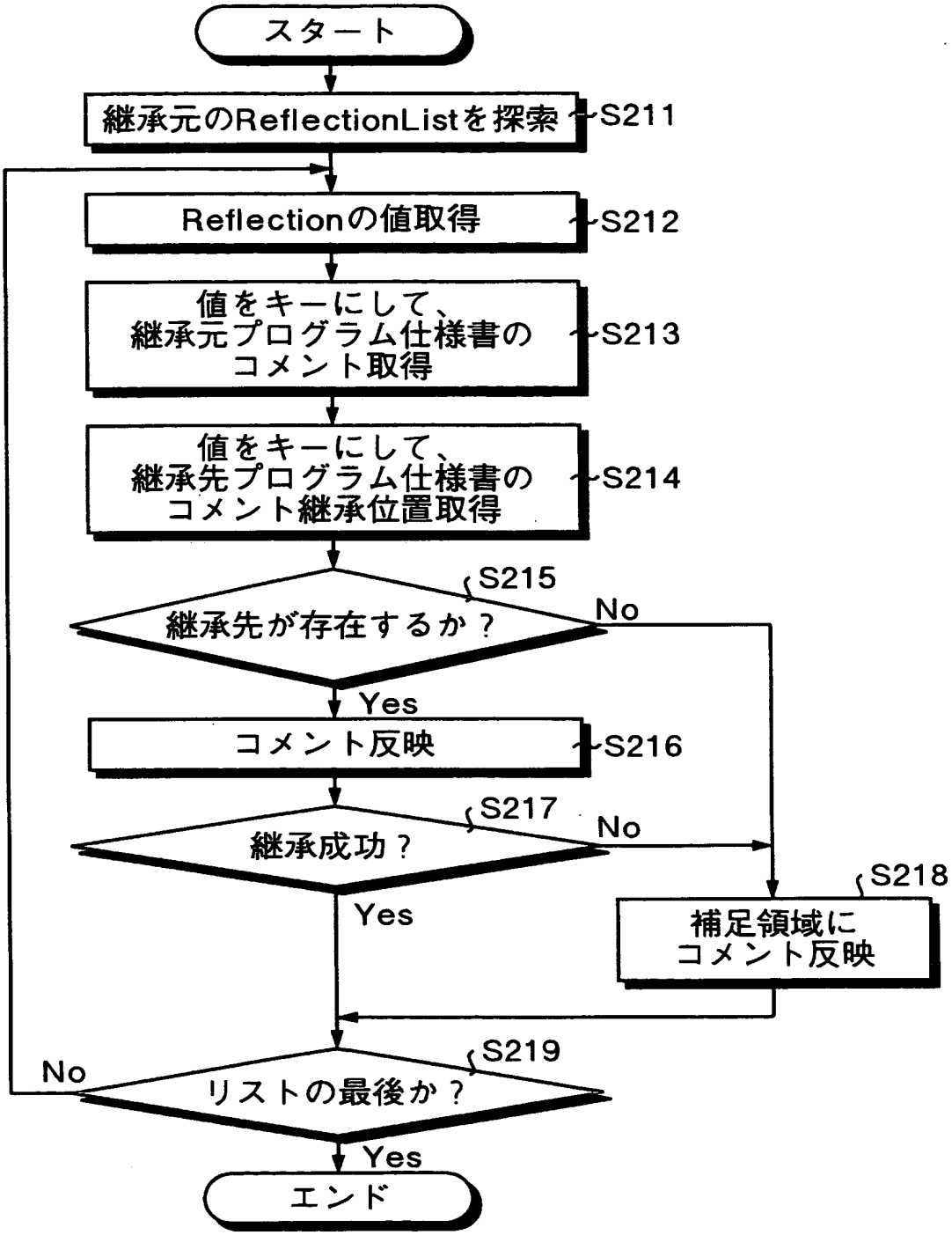
【図１６－２】

複数の記述領域がある場合のコメント記述領域に対応するプロパティのデータ構造の一例を示す図



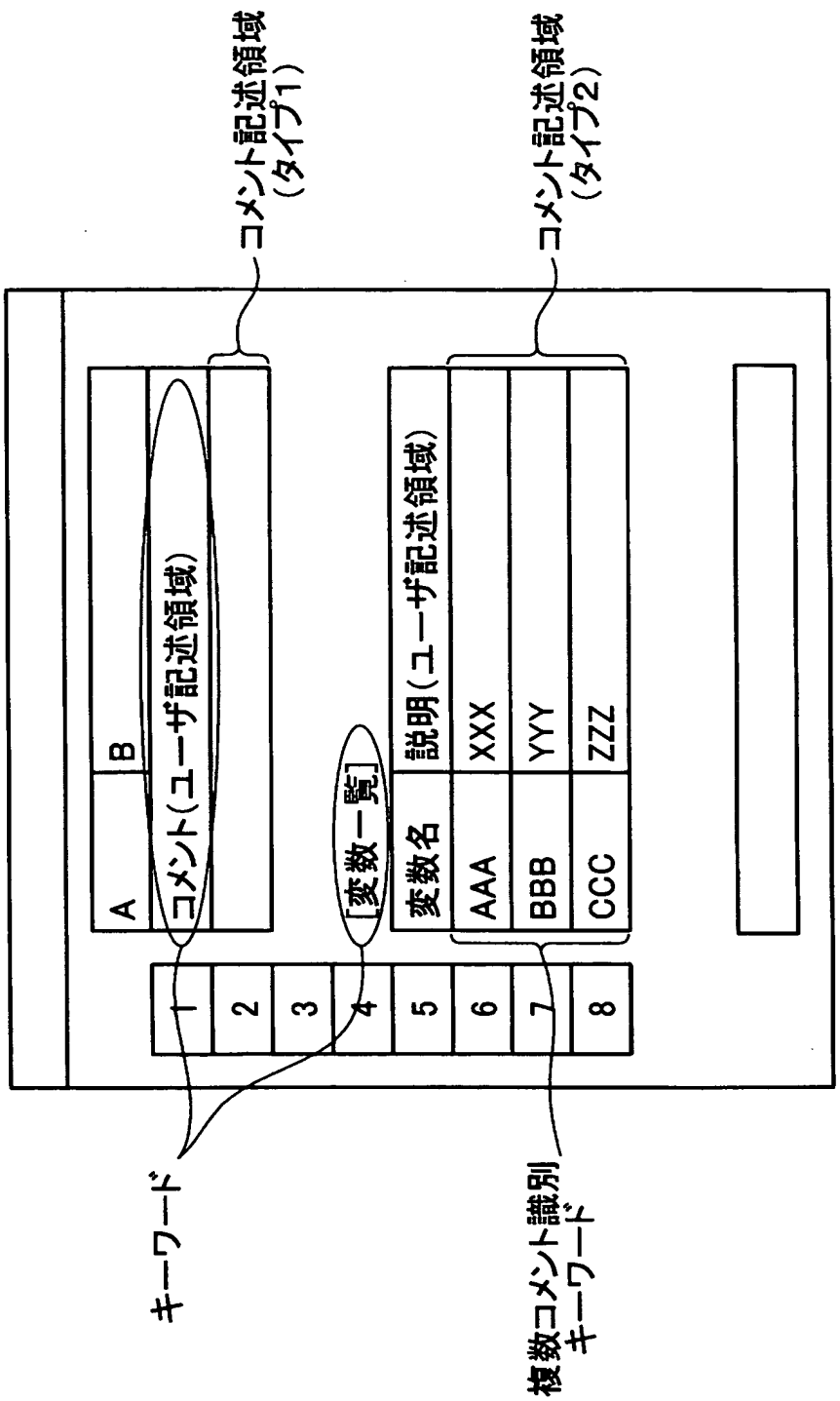
【図１７】

図１４に示したコメント記入内容継承部の
処理手順を示すフローチャート



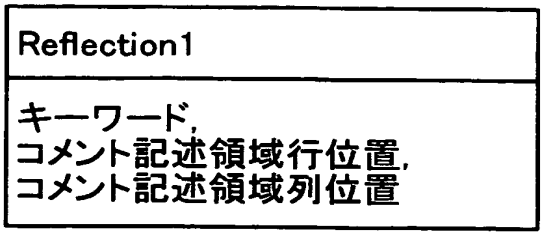
【図１８】

スプレッドシートを用いたプログラム仕様書のコメント記述領域の一例を示す図



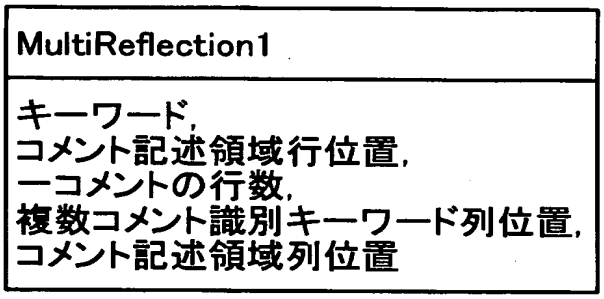
【図１９】

スプレッドシートを用いたプログラム仕様書のコメント記述領域に対応するプロパティのデータ構造の一例を示す図



【図２０】

スプレッドシートを用いたプログラム仕様書のコメント記述領域（複数の記述領域がある場合）に対応するプロパティのデータ構造の一例を示す図



【図２１】

処理構造図におけるコメント記述領域を示す図

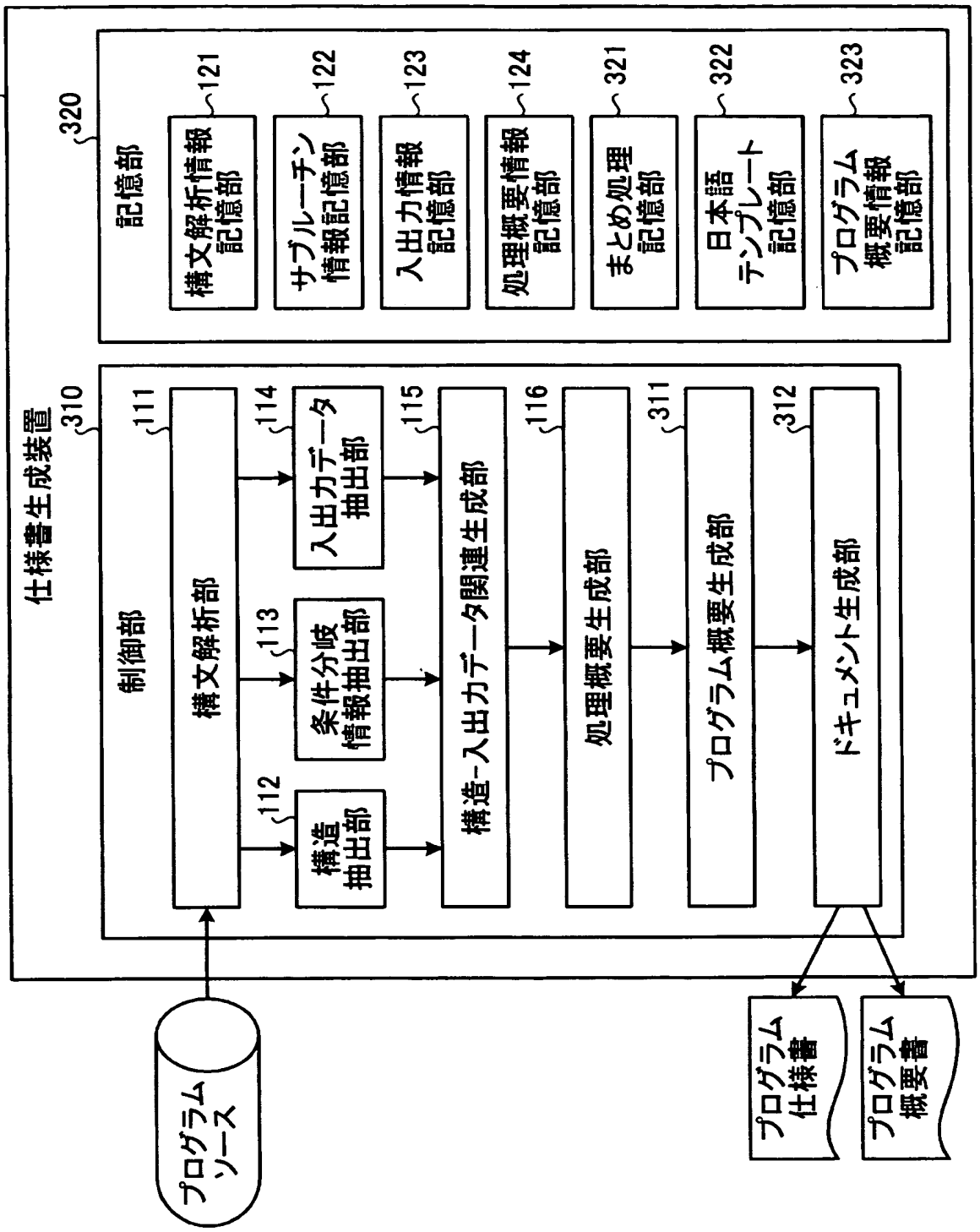
プログラム名	SECTION(1層目)	SECTION(2層目)	読み取りファイル	書き込みファイル	備考
プログラム		コメント(2)			
	初期処理				
コメント(1)					
	実行条件				
	UNTIL(終了フラグ＝定数－終了)				
	コメント(3)				
	繰り返し処理		in1(rec1)	out1(rec1)	
				out2(rec2)	
				out3(rec3)	
				out4(rec4)	
			コメント(5)	out5(rec5)	
		CALL "COM0002"			
		コメント(4)			
	繰り返し処理			コメント(6)	
		GOTO <loop>			
		<finish>			
		コメント(7)			
	終了処理				
					コメント(8)

図21に示した処理構造図における各コメント記述領域の位置情報データの一例を示す図

情報名→	セクション構造 （出現個数を意識した構造）	変位 （対応セクションから）	付加列名 （構造以外の欄の場合）
コメント (1)	プログラム名	行２、列０	
コメント (2)	プログラム名	行０、列２	
コメント (3)	プログラム名 └初期処理　[1]	行４、列０	
コメント (4)	プログラム名 └繰り返し処理　[1] └　CALL "COM0002" [1]	行１、列０	
コメント (5)	プログラム名 └繰り返し処理　[1]		読み取りファイル
コメント (6)	プログラム名 └繰り返し処理　[2]		書き込みファイル
コメント (7)	プログラム名 └繰り返し処理　[2] └ <finish> [1]	行２、列０	
コメント (8)	プログラム名 └終了処理　[1]		備考

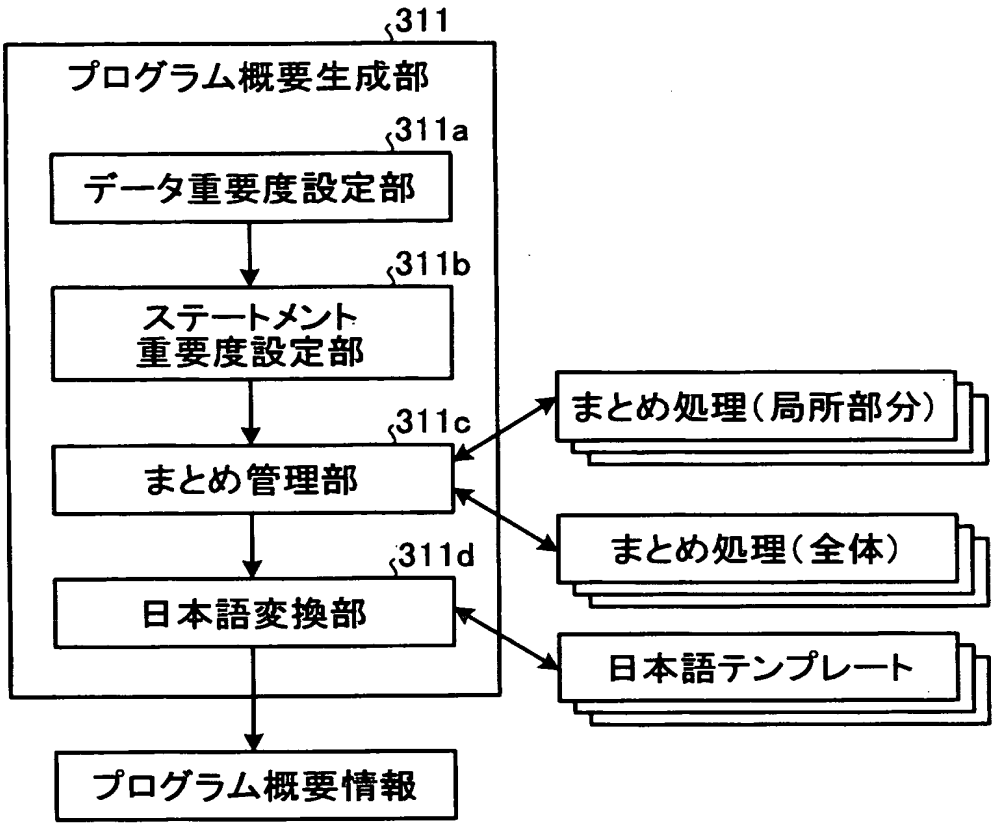
【図２３】

本実施例3に係る仕様書生成装置の構成を示す機能ブロック図



【図２４】

プログラム概要生成部の構成を示す機能ブロック図



【図２５－１】

プログラムソース例(COBOL)を示す図

```
MAIN                          SECTION.
MAIN-START.

    MOVE 0 TO W-ERR-FLAG.
    IF  IN-CODE  NOT  =  103
        GO TO   NEXT-DATA-READ
    END-IF.

    PERFORM   MASTER-MODIFY-SECT.
    IF  W-ERR-FLAG  NOT  =  0
        GO TO   MAIN-END
    END-IF.

NEXT-DATA-READ.
    PERFORM   FILE-READ-SECT.

MAIN-END.
    EXIT.
```

【図２５－２】

図25-1に示したプログラムソース例から生成されるプログラム中間情報の例を示す図

```
< section name="MAIN">
   <paragraph name="MAIN - START">
     <sequences num="4">
      <move>
           <ref><constant value="0" type="int"/></ref>
         <def><var name="W -ERR- FLAG"/></def>
      </move>
      <if>
          <condition><expression>
           <var name="IN - CODE"/>
             <comparison_operator name="NOT ="/>
             <constant value="103" type="int"/>
          </expression></condition>
        <then>
           <sequences num="1">
            <goto>
                <target type="paragraph"
                        name="NEXT -DATA - READ"
                         section_name="MAIN"/>
            </goto>
           </sequences>
         </then>
      </if>
       <perform_external>
          <target type="section" name="MASTER  - MODIFY - SECT"/>
       </perform_external>
      <if>
          <condition><expression>
           <var name="W -ERR- FLAG"/>
             <comparison_operator name="NOT ="/>
             <figurative_constant name="ZERO"/>
          </expression></condition>
        <then>
           <sequences num="1">
            <goto>
                <target type="paragraph"
                        name="MAIN - END"
                         section_name="MAIN"/>
            </goto>
           </sequences>
        </then>
      </if>
     </sequences>
   </paragraph>
    <paragraph name="NEXT -DATA - READ">
     <sequences num="1">
       <perform_external>
           <target type="section" name="FILE  - READ - SECT"/>
       </perform_external>
     </sequences>
   </paragraph>
    <paragraph name="MAIN - END">
     <sequences num="1">
       <exit_sentence/>
     </sequences>
   </paragraph>
</section>
```

【図２６】

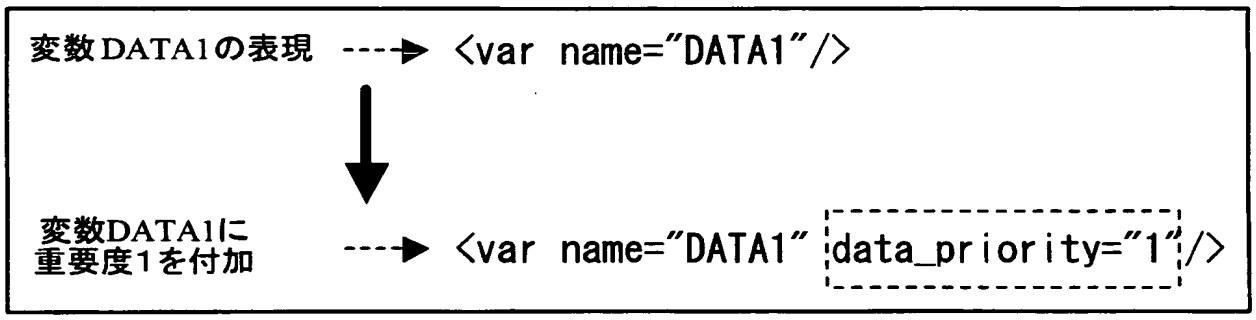
データの重要度の分類の一例を示す図

重要度	分類ID	説明
1	D-1	処理パスの分岐条件に関わるデータ
2	D-2	ファイル／データベースなどの出力に使われるデータ
3	D-3	ファイル／データベースなどからの入力に使われるデータ
4	D-4	その他のデータ

重要度高 ← → 重要度低

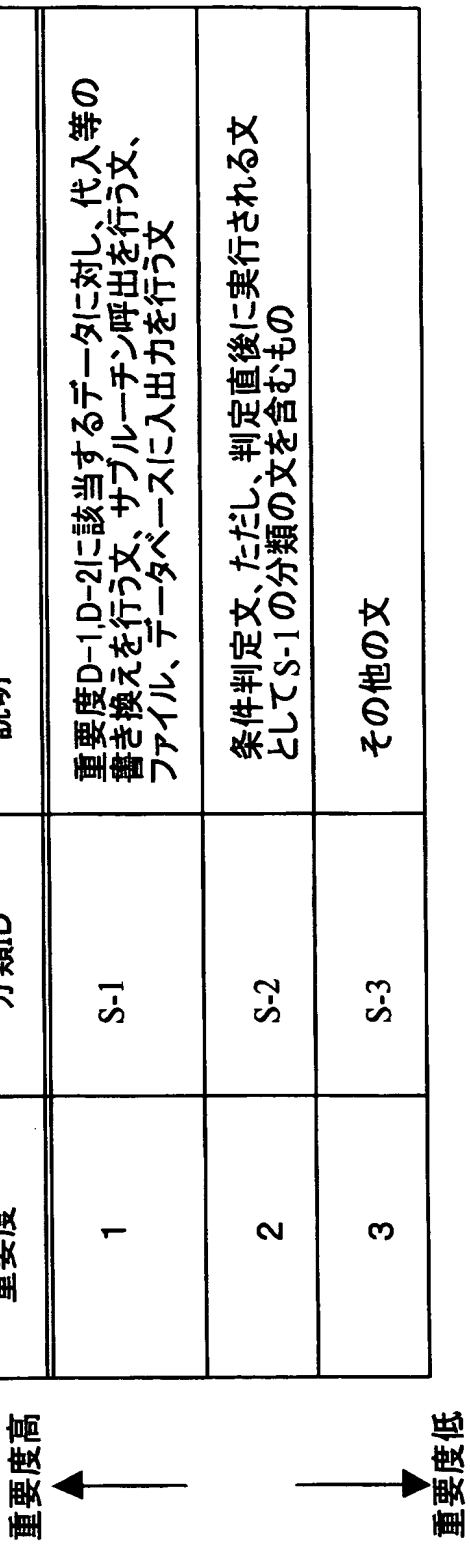
【図２７】

プログラム中間情報の変数用タグに付加される重要度の一例を示す図



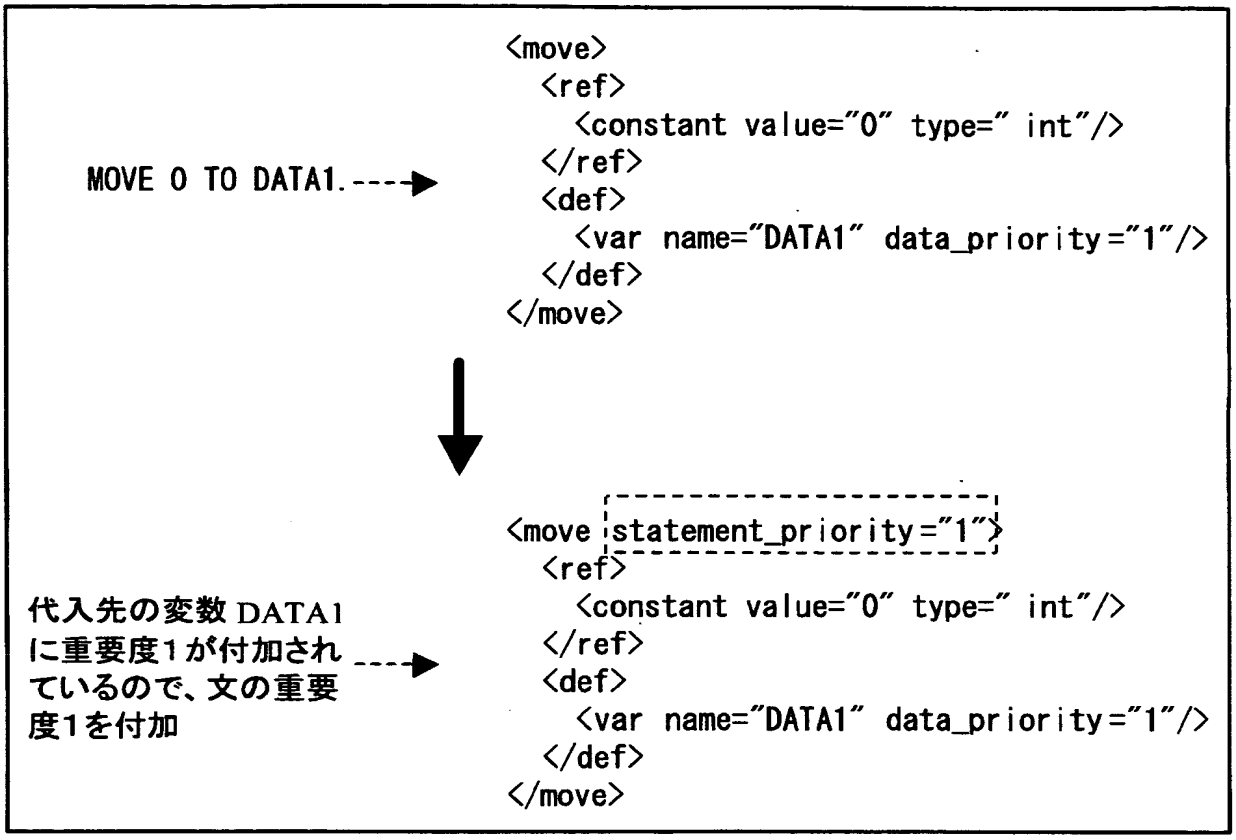
【図２８】

ステートメントの重要度の分類の一例を示す図



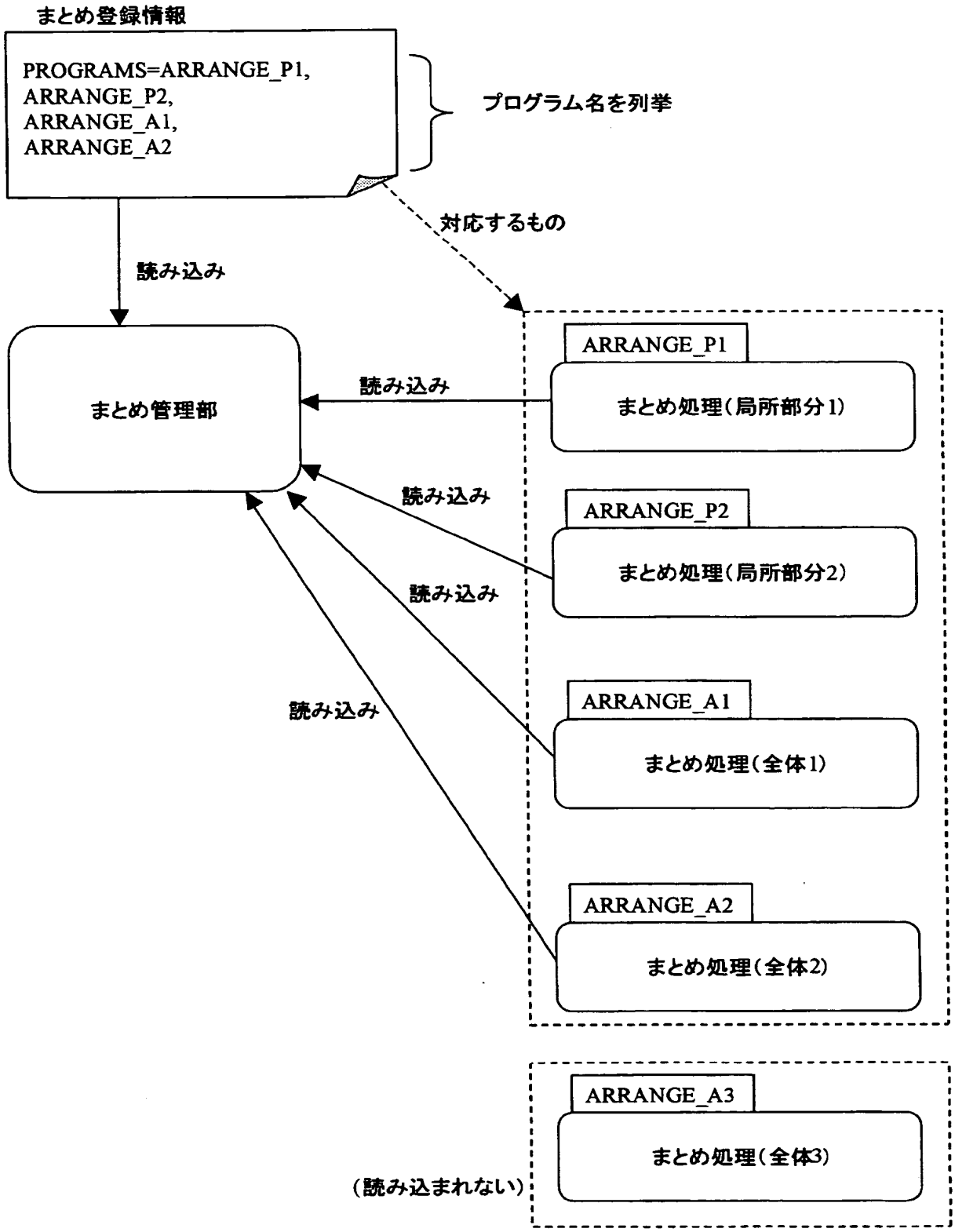
【図２９】

プログラム中間情報のステートメントのタグに付加される重要度の一例を示す図



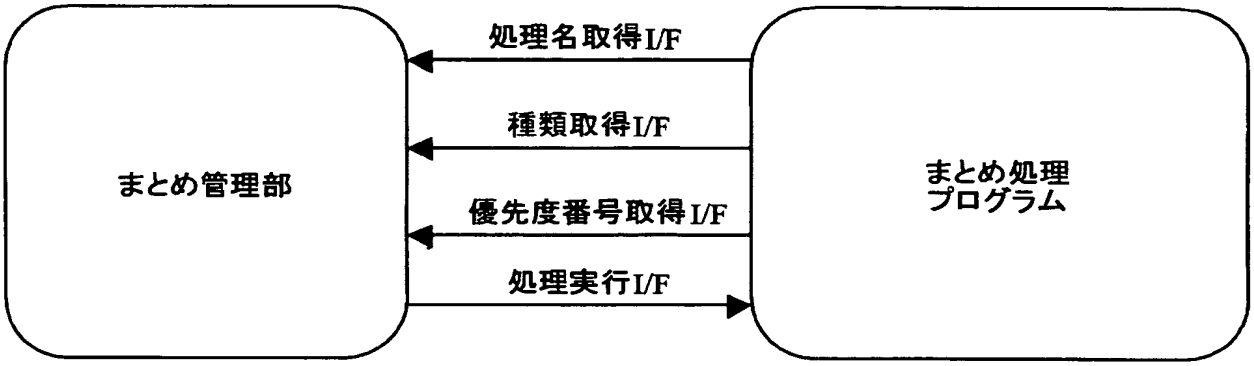
【図３０】

まとめ管理部の初期動作を説明するための説明図



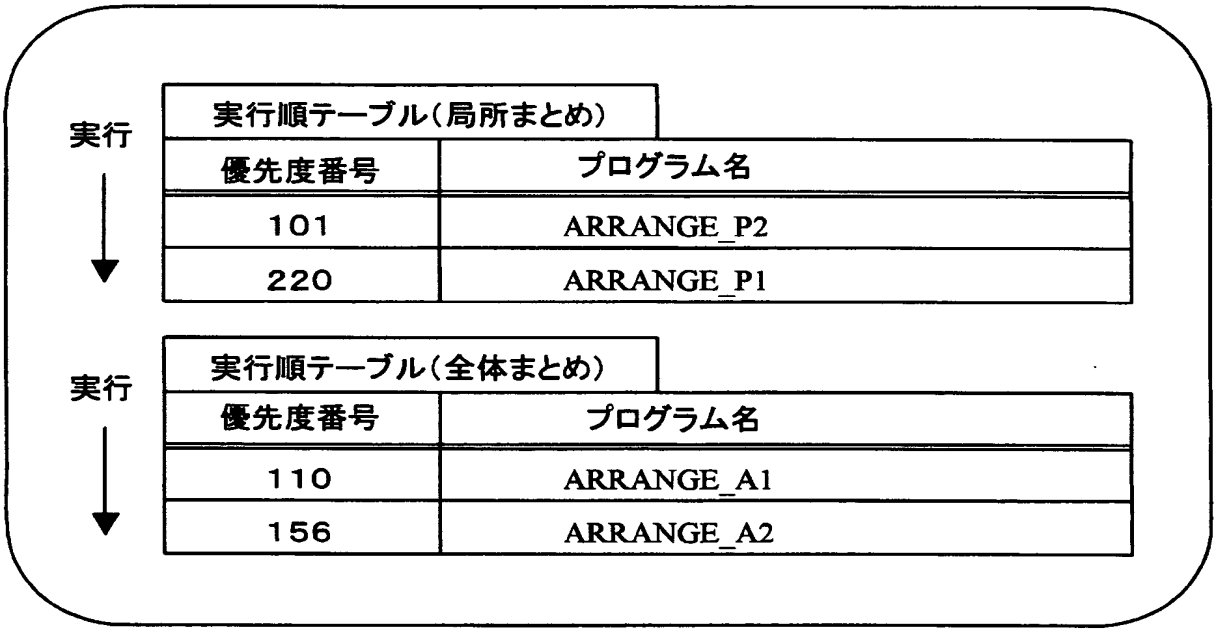
【図３１】

まとめ管理部とまとめ処理プログラムのインタフェース(I/F)を示す図



【図３２】

まとめ処理プログラムの実行順序保持の一例を示す図



【図３３－１】

まとめ処理プログラムが行うまとめのルール（局所的）の
例を示す図（その１）

プログラム名：AR_P1	処理名：READ-UNTIL のまとめ
種類：局所	優先度番号：100

説明：

READ 文のファイル終了時の処理が、UNTIL 条件の成立と一致している場合、UNTIL の条件を「ファイル～の終りまで」に置き換える。また、READ 文のファイル終了時の処理を削除する。

例：

```
READ INF AT END MOVE "END" TO FLAG .
PERFORM ～ UNTIL FLAG = "END".
```

↓

```
READ INF.
PERFORM ～ UNTIL AT_FILE_END INF.
```

<at_file_end >タグを処理の中では挿入する。

【図３３－２】

まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その2）

プログラム名：AR_P2	処理名：セクションの展開
種類：局所	優先度番号：200

説明：

セクションの内容が短く、３行までの場合でかつ、呼出元（上位セクション）にて無条件で呼ばれている場合、セクションの内容を上位のセクションに含む形に展開する。

例：

```
～
 PERFORM  READ -SUB.
～
READ-SUB          SECTION.
 READ INF .
 EXIT.
```

↓

```
～
 READ INF.
～
```

【図３３－３】

まとめ処理プログラムが行うまとめのルール（局所的）の
例を示す図（その３）

プログラム名：AR_P3	処理名：代入の集団項目化
種類：局所	優先度番号：300

説明：
MOVE文が連続し、代入元の項目がすべて同じ集団項目に所属し、また、代入先の項目がすべて同じ集団項目に所属するとき、それら集団項目名による代入文に置き換える。なお、集団項目内のすべての項目が列挙されていない場合は、その集団項目の部分であることを示す属性を加える。

例：

```
MOVE IN-DATA1 TO OUT -DATA1.
MOVE IN-DATA2 TO OUT -DATA1.
(IN-DATA1、IN-DATA2 が INR に属し、
OUT0DATA1、OUT-DATA2 が OUTRに属する)
```

↓

```
MOVE INR TO OUTR.
```

【図３３－４】

まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その4）

プログラム名：AR_P4	処理名：重要度の低い文の削除
種類：局所	優先度番号：400

説明：

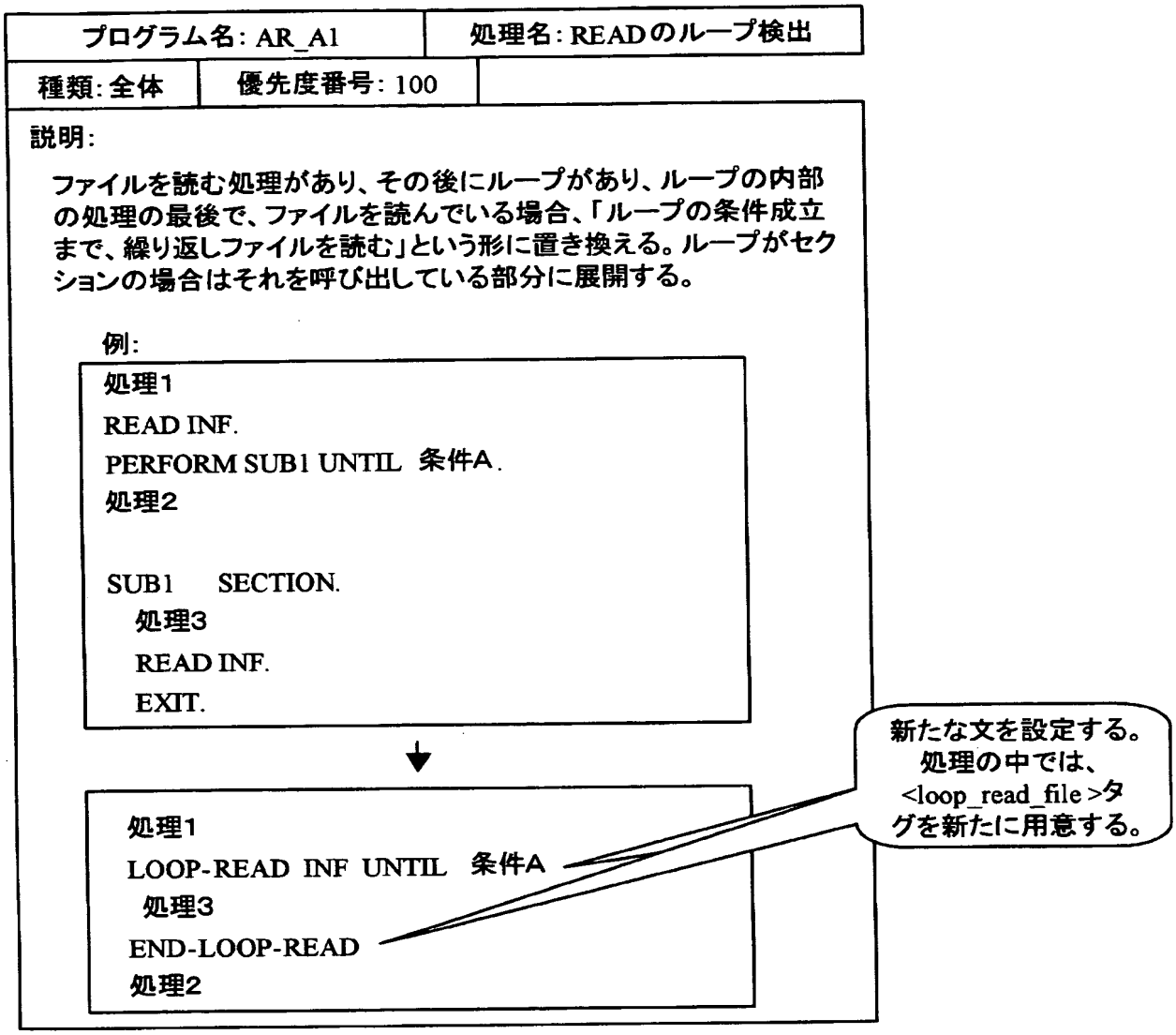
（１）statement_priority が"3"になっている文を削除する。また、このとき、削除される文のタグの親タグ <sequence> の属性 qt の値を削除した文の数だけ減らす。（ qt はその内部に含む文の数を示す）

（２）また、このとき、<sequence> の属性qtの値が0の場合、それを含むセクション、パラグラフまたは、条件文（ IF,EVALUATE, READ 後条件）を削除する。セクションの場合はその呼出元の呼出文(EVALUATE) も削除する。

なお、（１）、（２）は文の数が変わらなくなるまで対象となるプログラムについて繰り返し行う。

【図３３－５】

まとめ処理プログラムが行うまとめのルール（全体的）の例を示す図（その１）



【図３３－６】

まとめ処理プログラムが行うまとめのルール（全体的）の
例を示す図（その2）

プログラム名：AR_A2	処理名：変数の全体名記述
種類：全体	優先度番号：200

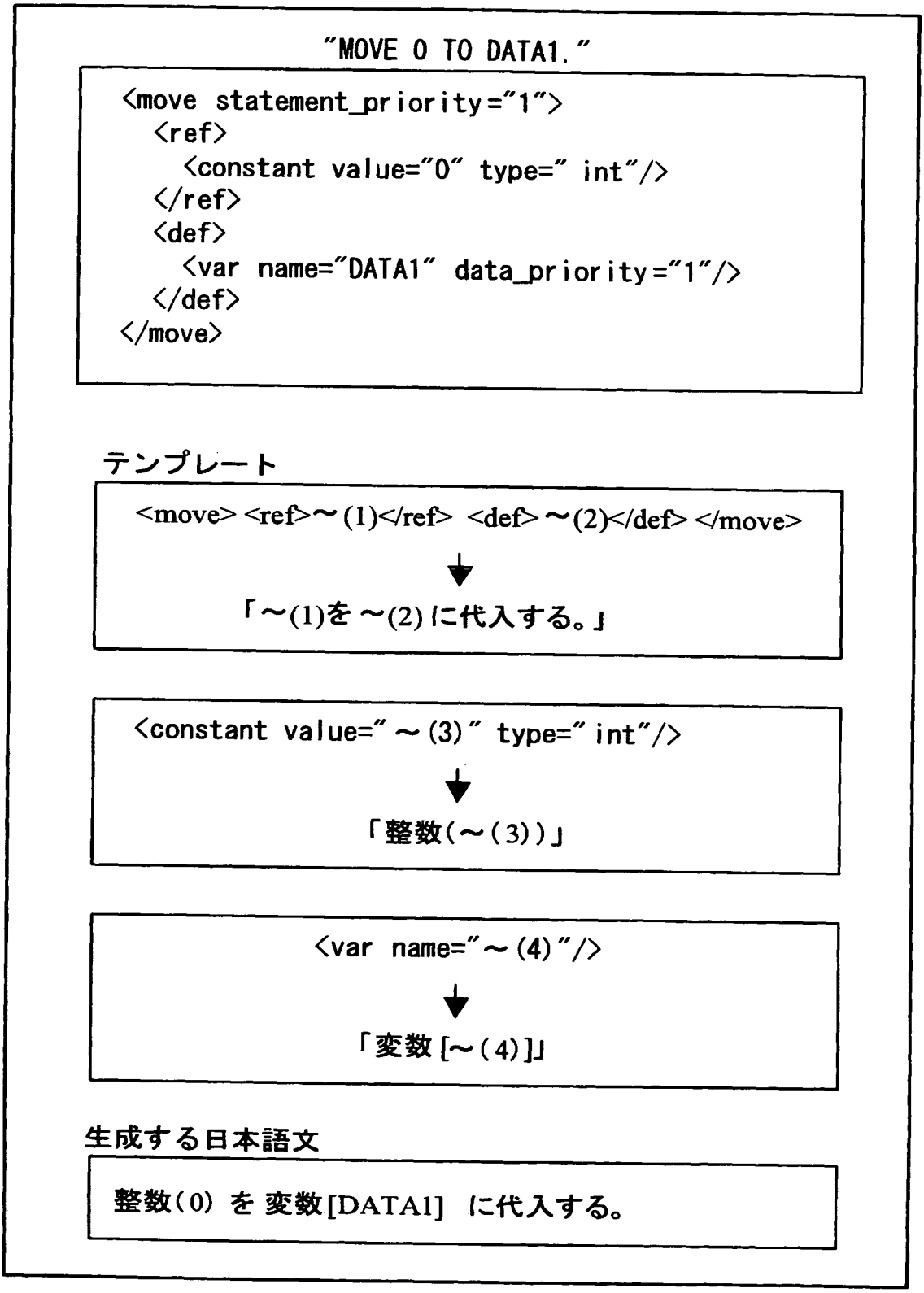
説明：

変数の項目名を集団項目名も付加した形で表現する。名前の間は"."で連結する。また、もっとも上位の集団項目名から表現する。

IN-RECORD.IN-DATA1 などの形式。

【図３４】

日本語変換部の処理を説明するための説明図



【図３５－１】

その他の日本語テンプレートの例を示す図（その１）

```
<loop_read_file>
  <file name="～(1)">
    <record>～(2)</record>
  </file>
  <until>～(3)</until>
  <sequences>～(4)</sequences>
</loop_read_file>
```

↓

ファイル～(1)から～(2)へレコードを読み込み、～(3)まで繰り返し行う。
読むたびに以下を行う。

「～(4)」

```
<if>
  <condition>～(1)</condition>
  <then><sequence>～(2)</sequence></then>
  <else><sequence>～(3)</sequence></else>
</if>
```

↓

もし、条件～(1)が、成立する場合、以下を行う。

「～(2)」

もし、条件～(1)が、成立しない場合、以下を行う。

「～(3)」

```
<write>
  <file name="～(1)">
    <record>～(2)</record>
  </file>
</write>
```

↓

レコード～(2)をファイル～(1)に書き出す。

【図３５－２】

その他の日本語テンプレートの例を示す図（その2）

```
<perform_internal >
  <condition>
    <varying> ～ (1)</varying>
    <varying_from >～ (2)</ varying_from >
    <by>～ (3)</by>
    <until>～ (4)</until>
  </condition>
  <sequences> ～ (5)</sequences>
</ perform_internal >
```

↓

条件～(4)が成立するまで、以下を繰り返し実行する。このとき～(1)は～(2)から～(3)ずつ増やしていく。

「～(5)」

```
<move><ref part="on"> ～ (1)</ref><def> ～ (2)</def></move>
```

↓

「～(1)の一部を ～(2) に代入する。」

```
<at_end_file  name="～(1)"/>
```

↓

ファイル～(1)の終わり

```
<expression> ～ (1)</expression>
```

↓

式[～(1)]

```
<comparison_operator  name="～(1)"/>
```

↓

～(1)

```
<constant value="～(1)"/>
```

↓

～(1)

【図３６】

プログラムサンプルを示す図

```
 IDENTIFICATION    DIVISION.
PROGRAM-ID.        TESTSAMPLE.
 AUTHOR.           TARO.YAMADA.
 ENVIRONMENT       DIVISION.
 CONFIGURATION     SECTION.
SOURCE-COMPUTER.   VIRTUAL/HOST1.
OBJECT-COMPUTER.   VIRTUAL/HOST1.

 ********** IN OUT **********
INPUT-OUTPUT       SECTION.
FILE-CONTROL.
     SELECT INFILE    ASSIGN TO INFILE.
     SELECT OUTFILE   ASSIGN TO OUTFILE.

 ********** DATA DIVISION **********
DATA DIVISION.
FILE SECTION.

 ********** INPUT FILE **********
 FD  INFILE     BLOCK 0 RECORDS.
01  IN-RECORD.
     03  MEMBERCODE          PIC 9(5).
     03  JOB-CODE            PIC 9(3).
     03  NAME.
         05  NAMEFAMLY       PIC N(16).
         05  NAMEFIRST       PIC X(16).
     03  SEIBETSU            PIC 9.
     03  YUUBINBANGO.
         05  YUBIGCODE       PIC 9(3).
         05  YUSMALLCODE.    PIC 9(4).
     03  TEL                 PIC X(14).
     03  BIRTHDAY.
         05  BD-YEAR         PIC 9(4).
         05  BD-MONTH        PIC 9(2).
         05  BD-DAY          PIC 9(2).
     03  FAMILYDATA.
         05  F-MEMBER        OCCURS 20.
             07  F-TYPE      PIC 9(2).
             07  F-MEMBER-F  PIC 9.
             07  F-CODE      PIC 9(5).
      03  FILLER             PIC X(25).

 ********** OUTPUT FILE **********
 FD  OUTFILE     BLOCK 0 RECORDS.
01  OUT-RECORD.
     03 OUT-MEMBERCODE       PIC 9(5).
     03 OUT-F-TYPE           PIC 9(2).
     03 OUT-F-MEM-FLAG       PIC 9.
     03 OUT-F-CODE           PIC 9(5).
 ********** WORKING STORAGE **********
WORKING-STORAGE SECTION.
01  COUNTER-TABLE.
     03 IN-COUNT             PIC S9(9)  VALUE ZERO.
     03 SKIP-COUNT           PIC S9(9)  VALUE ZERO.
     03 PROC-COUNT           PIC S9(9)  VALUE ZERO.
     03 OUT-COUNT            PIC S9(9)  VALUE ZERO.
 01  I                       PIC S9(4).
01  END-FLAG                 PIC X(03)  VALUE SPACE.
```

```
********** PROCEDURE DIVISION **********
PROCEDURE DIVISION.

     OPEN    INPUT   INFILE     OUTPUT  OUTFILE.
    DISPLAY "## プログラムスタート" UPON CONSOLE.

    PERFORM   READ-SECT.
    IF END-FLAG  = "END"
       DISPLAY "## 一つもレコードを読めません。" UPON CONSOLE
       DISPLAY "## エラー終了しました。"         UPON CONSOLE
       STOP RUN                                          [1]
    END-IF.

    PERFORM   MAIN-SECT  UNTIL   END-FLAG = "END"

    CLOSE   INFILE     OUTFILE.
    DISPLAY "○出力件数        =[" OUT-COUNT  "]" UPON CONSOLE.
    DISPLAY "正常終了しました。"      UPON CONSOLE.

    STOP RUN.

 ********** READ ROUTINE **********
READ-SECT    SECTION.
    READ    INFILE
      AT END        MOVE  "END" TO   END-FLAG
      NOT AT END    ADD   1     TO   IN-COUNT
    END-READ.                                          [2]
READ-SECT-END.
    EXIT.

 ********** MAIN ROUTINE **********
MAIN-SECT    SECTION.
     IF FAMILYDATA = ZERO
        ADD     1       TO      SKIP-COUNT
     ELSE
        ADD     1       TO      PROC-COUNT
        PERFORM VARYING I FROM 1 BY 1
                         UNTIL I > 20
           MOVE    MEMBERCODE    TO  OUT-MEMBERCODE
           MOVE    F-TYPE(I)     TO  OUT-F-TYPE
[3]        MOVE    F-MEMBER-F(I) TO  OUT-F-MEM-FLAG
           MOVE    F-CODE(I)     TO  OUT-F-CODE
           WRITE   OUT-RECORD
           ADD     1             TO  OUT-COUNT
        END-PERFORM
    END-IF.

    PERFORM   READ-SECT.

MAIN-SECT-END.
    EXIT.
```

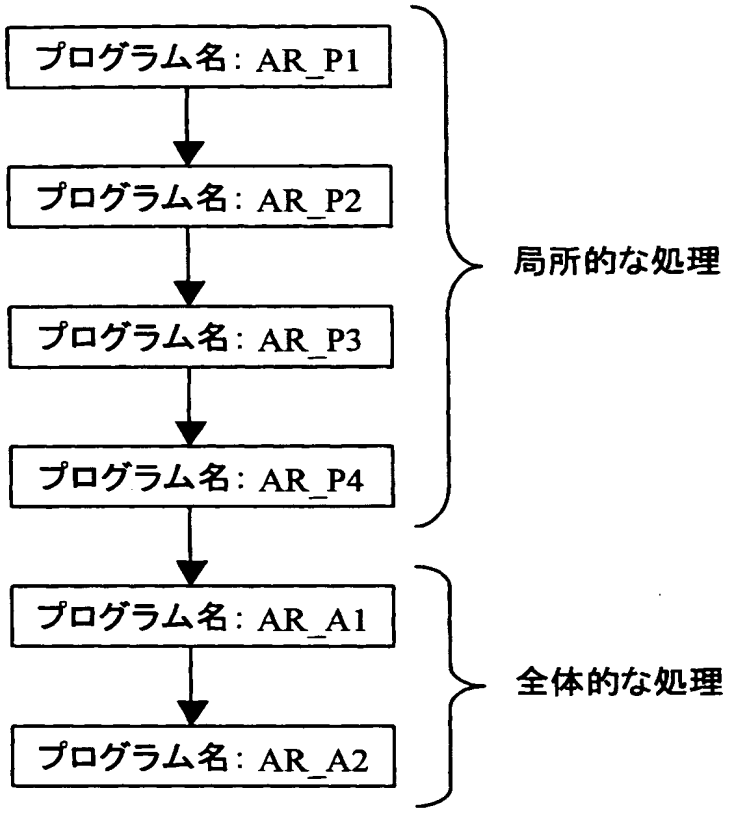
【図３７】

まとめ管理部が利用するまとめ処理プログラムのリストを示す図

プログラム名：AR_P1	処理名：READ-UNTILのまとめ	種類：局所	優先度番号：100
プログラム名：AR_P2	処理名：セクションの展開	種類：局所	優先度番号：200
プログラム名：AR_P3	処理名：代入の集団項目化	種類：局所	優先度番号：300
プログラム名：AR_P4	処理名：重要度の低い文の削除	種類：局所	優先度番号：400
プログラム名：AR_A1	処理名：READのループ検出	種類：全体	優先度番号：100
プログラム名：AR_A2	処理名：変数の全体名記述	種類：全体	優先度番号：200

【図３８】

まとめ処理プログラムの実行順序を示す図



【図３９】

まとめ処理終了時のプログラム中間情報を示す図

```
< loop_read_file statement_priority="1">
  <file name="INFILE">
    <record><var name="IN-RECORD" data_priority="3"/></record>
  </file>
  <until>< at_end_file  name="INFILE"/></until>
  <sequences qt="2">
    <if  statement_priority="2">
      <condition>
        <expression>
          <var  name="IN- RECORD.FAMILYDATA" data_priority="1"/>
          < comparison_operator  name="="/>
           <constant value="ZERO"/>
        </expression>
      </condition>
      <else>
        <sequences qt="2">
          < perform_internal statement_priority="1">
            <condition>
              <varying><var name="I" data_priority="1"/></varying>
              < varying_from ><constant value="1" type="int"/></ varying_from>
               <by><constant value="1" type="int"/></by>
              <until>
                <expression>
                  <var name="I" data_priority="1"/>
                  < comparison_operator  name="&gt;"/>
                   <constant value="20" type="int"/>
                </expression>
              </until>
            </condition>
            <sequences qt="2">
              <move  statement_priority="1">
                 <ref part="on" ><var name="IN-RECORD" data_priority="3"/></ref>
                <def><var name="OUT-RECORD" data_priority="2"/></def>
              </move>
              <write  statement_priority="1">
                 <file name="OUTFILE">
                   <record><var name="OUT-RECORD" data_priority="2"/></record>
                </file>
              </write>
            </sequences>
          </ perform_internal>
        </sequences>
      </else>
    </if>
  </sequences>
</ loop_read_file>
```

日本語変換部が生成した日本語情報を示す図

ファイルINFILEから変数IN-RECORDへレコードを読み込み、ファイル INFILEの終わりまで繰り返し行う。ファイルを読むたびに以下を行う。

「もし、条件式[変数IN-RECORD.FAMILYDATA=ZERO] 」が、成立しない場合、以下を行う。

「条件式 [変数I>20]が成立するまで、以下を繰り返し実行する。このとき変数 Iは1から1ずつ増やしていく。

「変数 IN-RECORDの一部を変数 OUT-RECORDに代入する。

レコード変数 OUT-RECORDをファイルOUTFILEに書き出す。

」

」

」

【図４１】

ドキュメント生成部が生成するプログラム
概要書の一例を示す図

プログラム概要書	作成日時
	2003/07/11 13:45

プログラム名	TESTSAMPLE	ファイル名	TESTSAMPLE.COB

コメント欄

ファイル情報

No.	ファイル名	外部名	種類
1.	INFILE	INFILE	COBOL ファイル
2.	OUTFILE	OUTFILE	COBOL ファイル

プログラム概要

ファイルINFILEから変数IN-RECORDへレコードを読み込み、ファイル INFILEの終わりまで繰り返し行う。ファイルを読むたびに以下を行う。

「もし、条件式[変数IN-RECORD.FAMILYDATA=ZERO]」が、成立しない場合、以下を行う。

「条件式[変数I>20]が成立するまで、以下を繰り返し実行する。このとき変数 Iは1から1ずつ増やしていく。

「変数IN-RECORDの一部を変数OUT-RECORDに代入する。

レコード変数OUT-RECORDをファイルOUTFILEに書き出す。

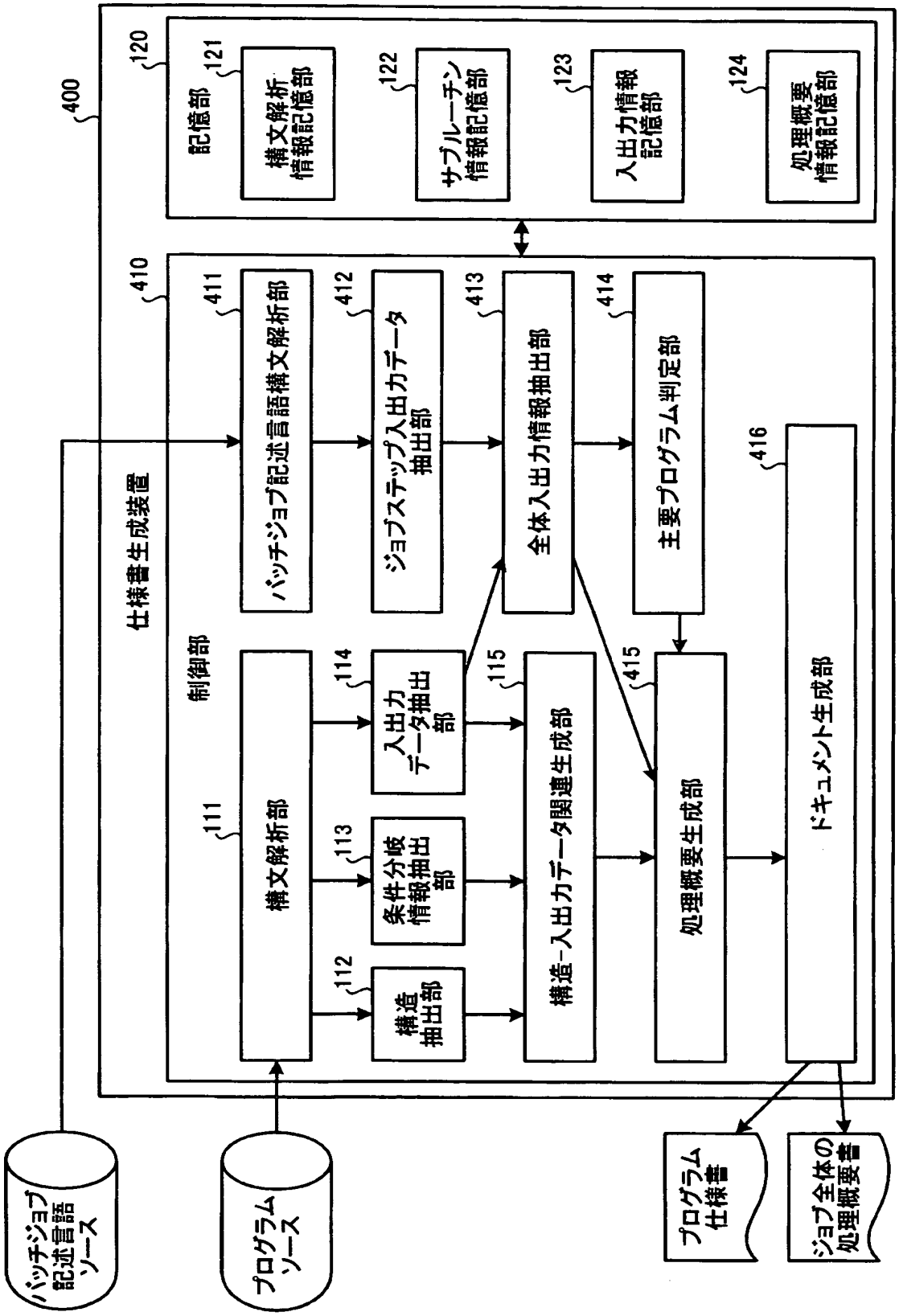
」

」

」

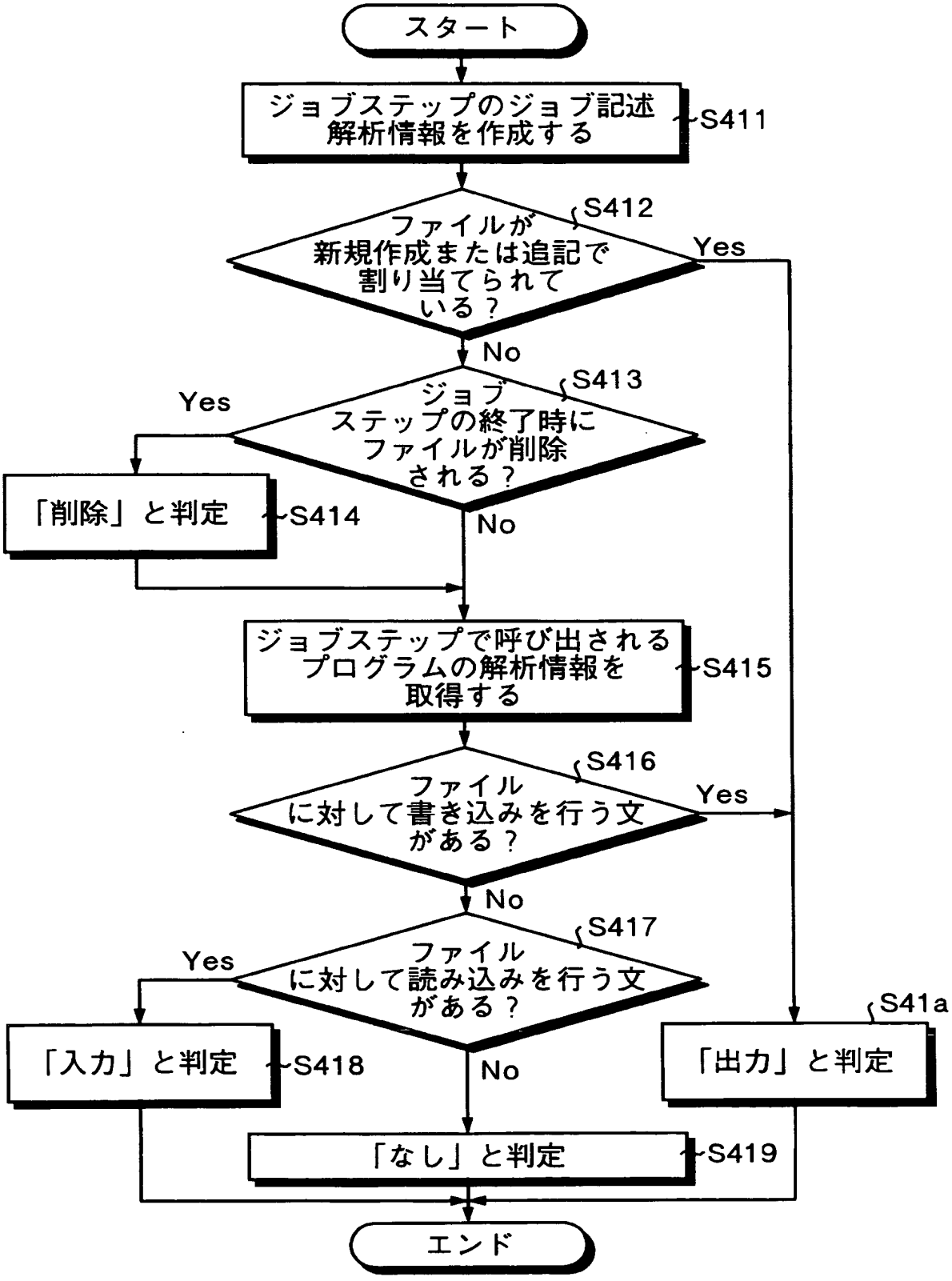
【図４２】

本実施例4に係る仕様書生成装置の構成を示す機能ブロック図



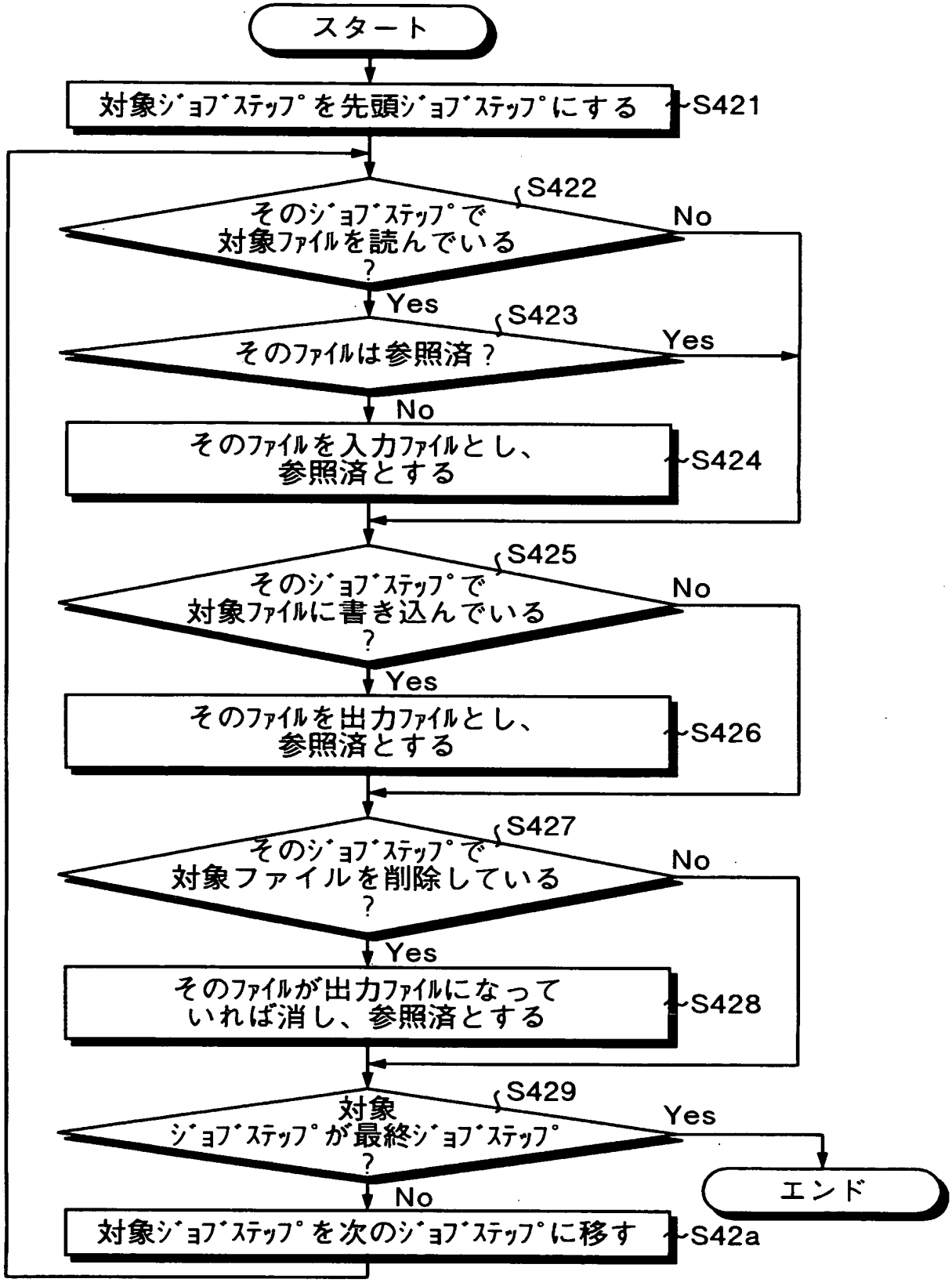
【図４３】

ジョブステップ入出力データ抽出部による入出力判定処理の処理手順を示すフローチャート



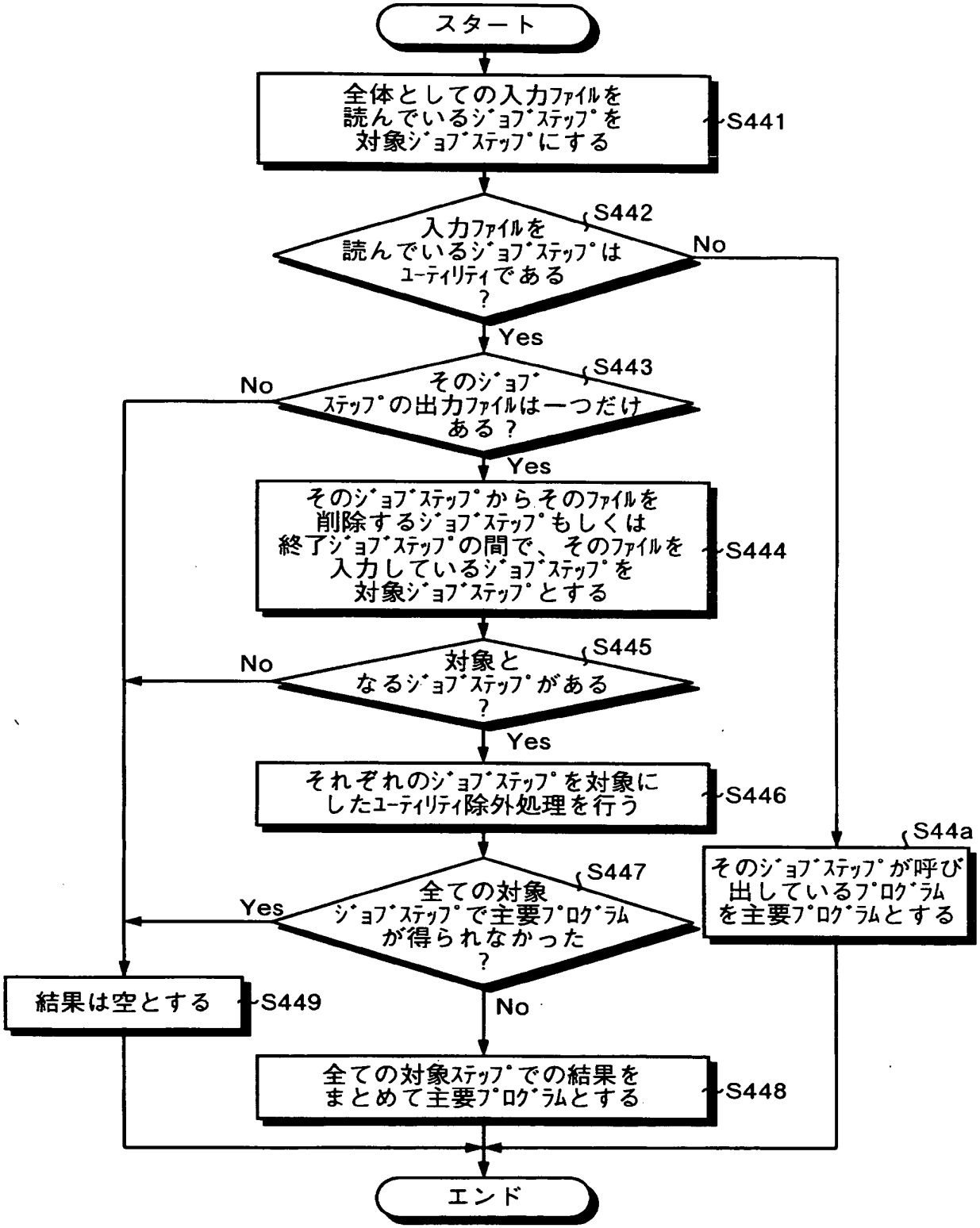
【図４４】

全体入出力情報抽出部によるジョブ全体としての
ファイル入出力判定処理の処理手順を示すフローチャート



【図４５】

ユーティリティ除外処理の処理手順を示すフローチャート



【図４６】

バッチジョブ記述の一例を示す図

```
///JOB01 JOB
//*****************************************
//STEP1 EXEC PGM=PROGA
//IN01  DD DSN=DENPYO01,DISP=SHR
//OT01  DD DSN=WORK1,DISP=(NEW,PASS)
//*****************************************
//STEP2 EXEC PGM=SORT
//SORTIN  DD DSN=WORK1,DISP=(OLD,DELETE)
//SORTOT  DD DSN=WORK2,DISP=(NEW,PASS)
//*****************************************
//STEP3 EXEC PGM=PROGB
//IN01  DD DSN=WORK2,DISP=(OLD,DELETE)
//OT01  DD DSN=SYUKEI1,DISP(NEW,PASS)
//*****************************************
//STEP4 EXEC PGM=ENDMSG
```

【図４７】

図46に示したバッチジョブ記述からバッジジョブ記述言語構造解析部が生成するバッチジョブ解析情報

ステップ名	呼び出しプログラム名	ファイル									
		DEPY01		WORK1		WORK2		DATA1			
		外部名	モード	外部名	モード	外部名	モード	外部名	モード		
STEP1	PROGA	IN01	S	OT01	NP						
STEP2	SORT			SOTRIN	OD	SORTOT	NP				
STEP3	PROGB					IN01	OD	OT01	NP		
STEP4	ENDMSG										

【図４８】

図46に示したバッチジョブ記述から生成されるバッチジョブ全体の処理概要を示す図

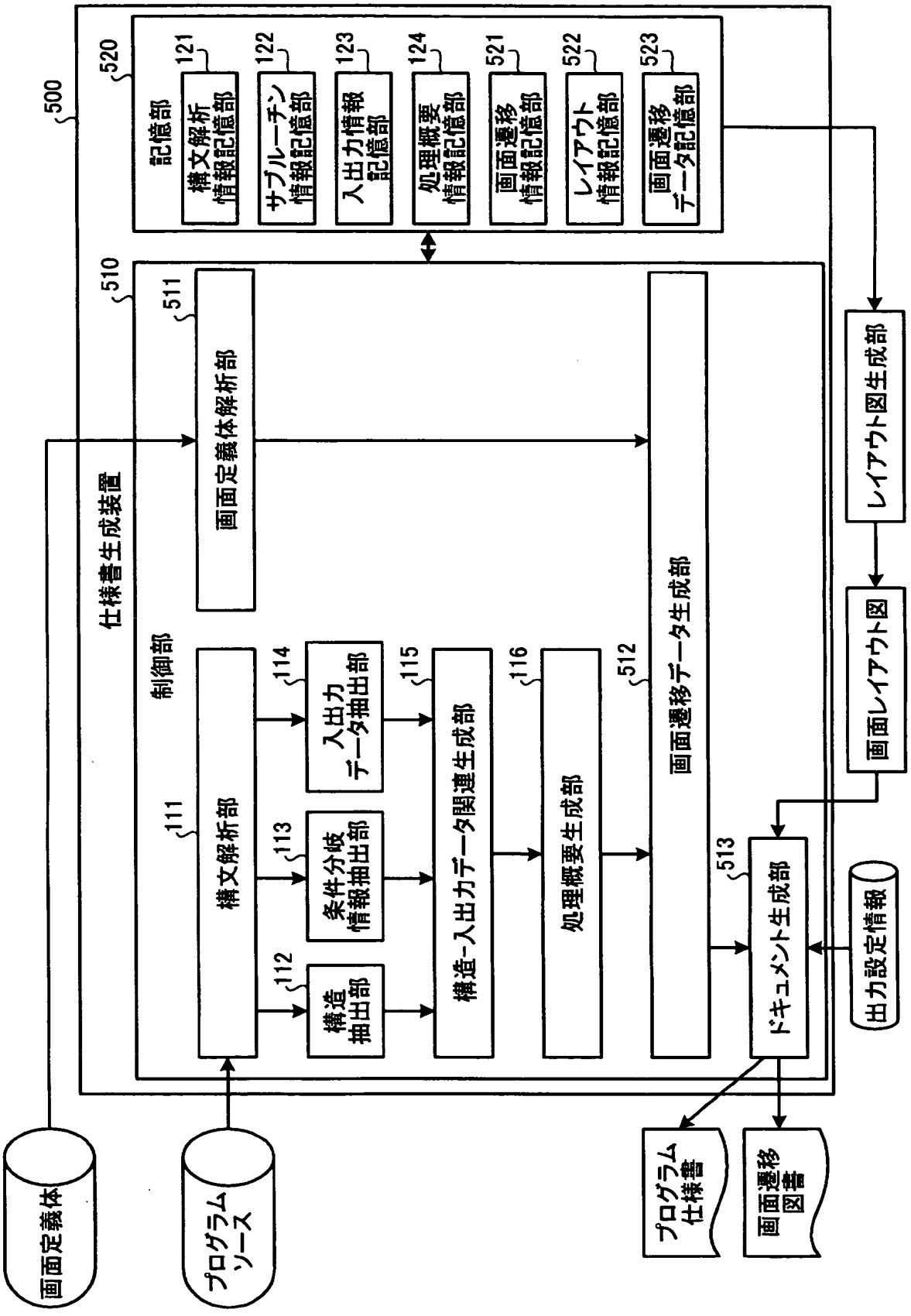
[入力]
DENPYO1

[出力]
SYUKEI1

[主要プログラム]
PROGA(新規伝票抽出処理),PROGB(当日集計処理)

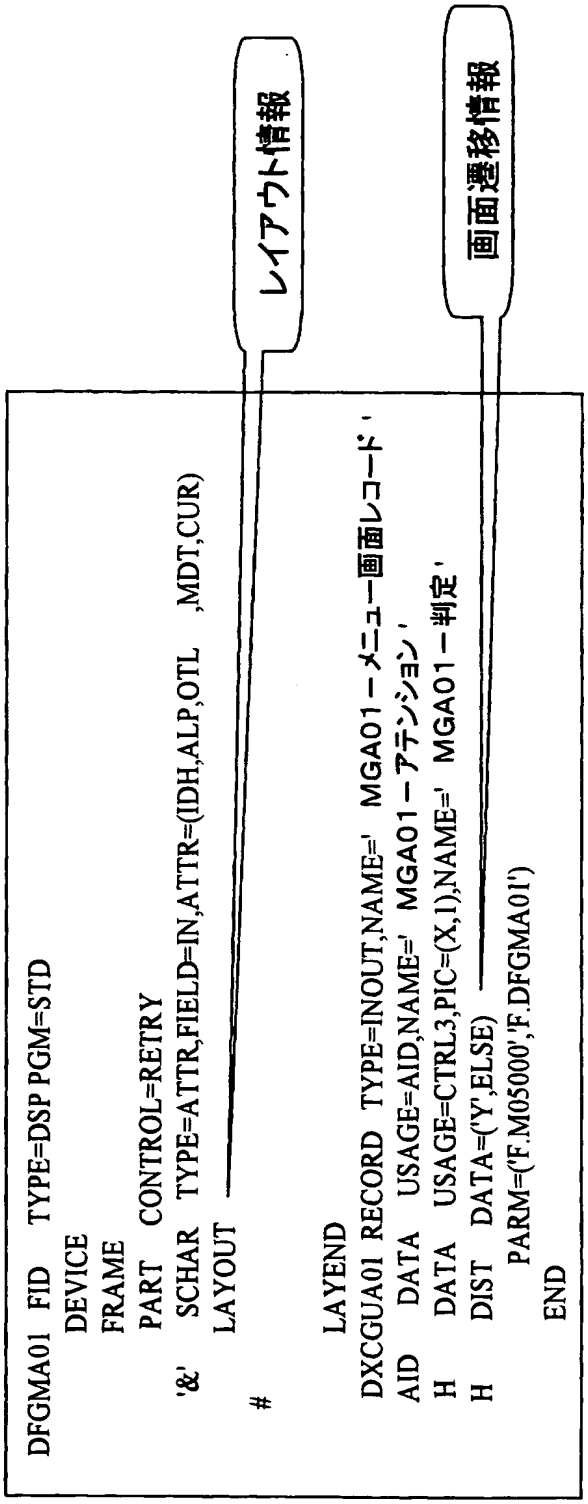
【図４９】

本実施例5に係る仕様書生成装置の構成を示す機能ブロック図



【図５０】

画面定義体の一例を示す図



【図５１】

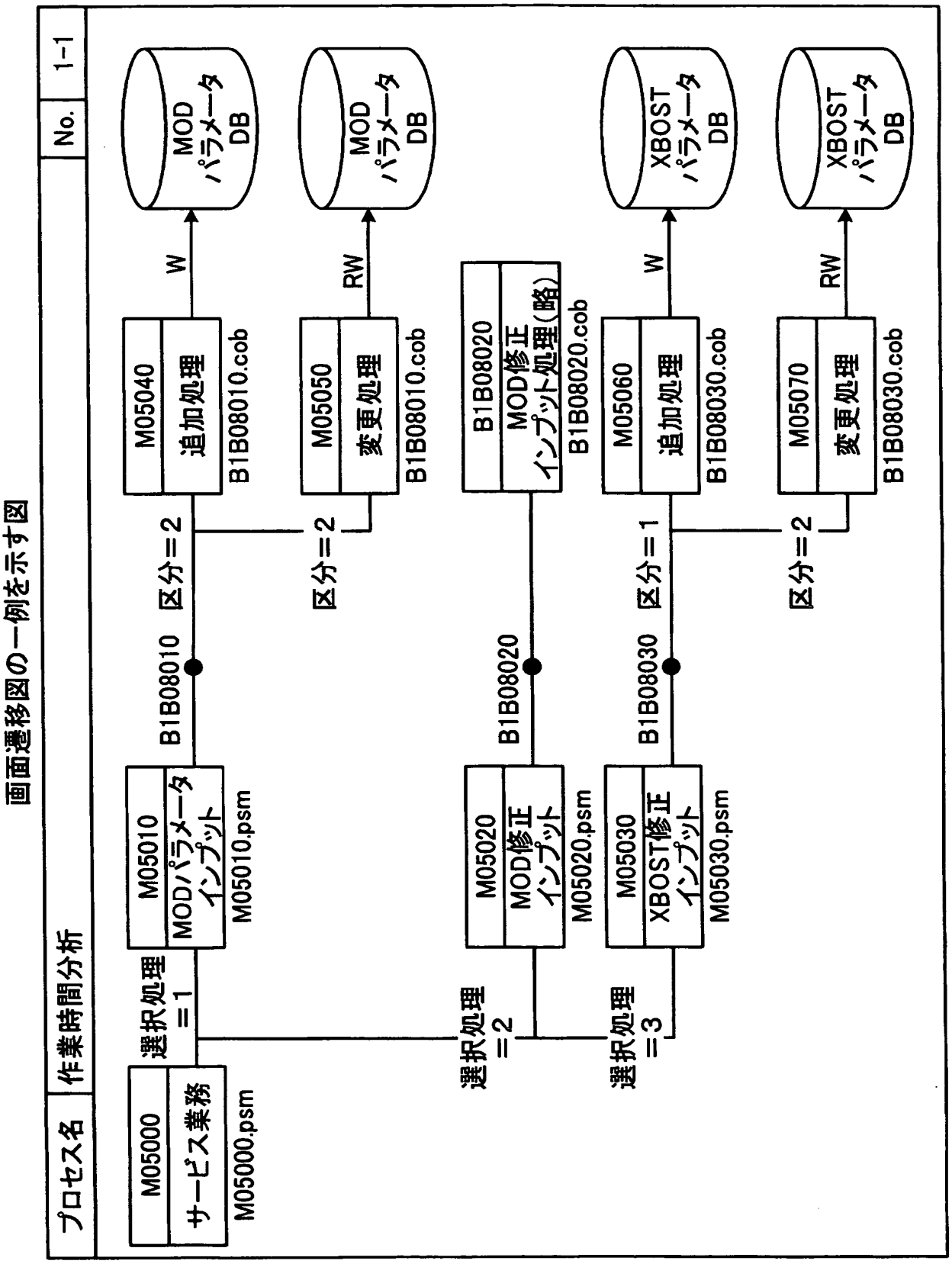
画面遷移情報の一例を示す図

遷移元画面	遷移条件	遷移先画面	遷移先種別
M05000	選択処理 =1	M05010	画面
M05000	選択処理 =2	PG0001	プログラム

画面遷移データの一例を示す図

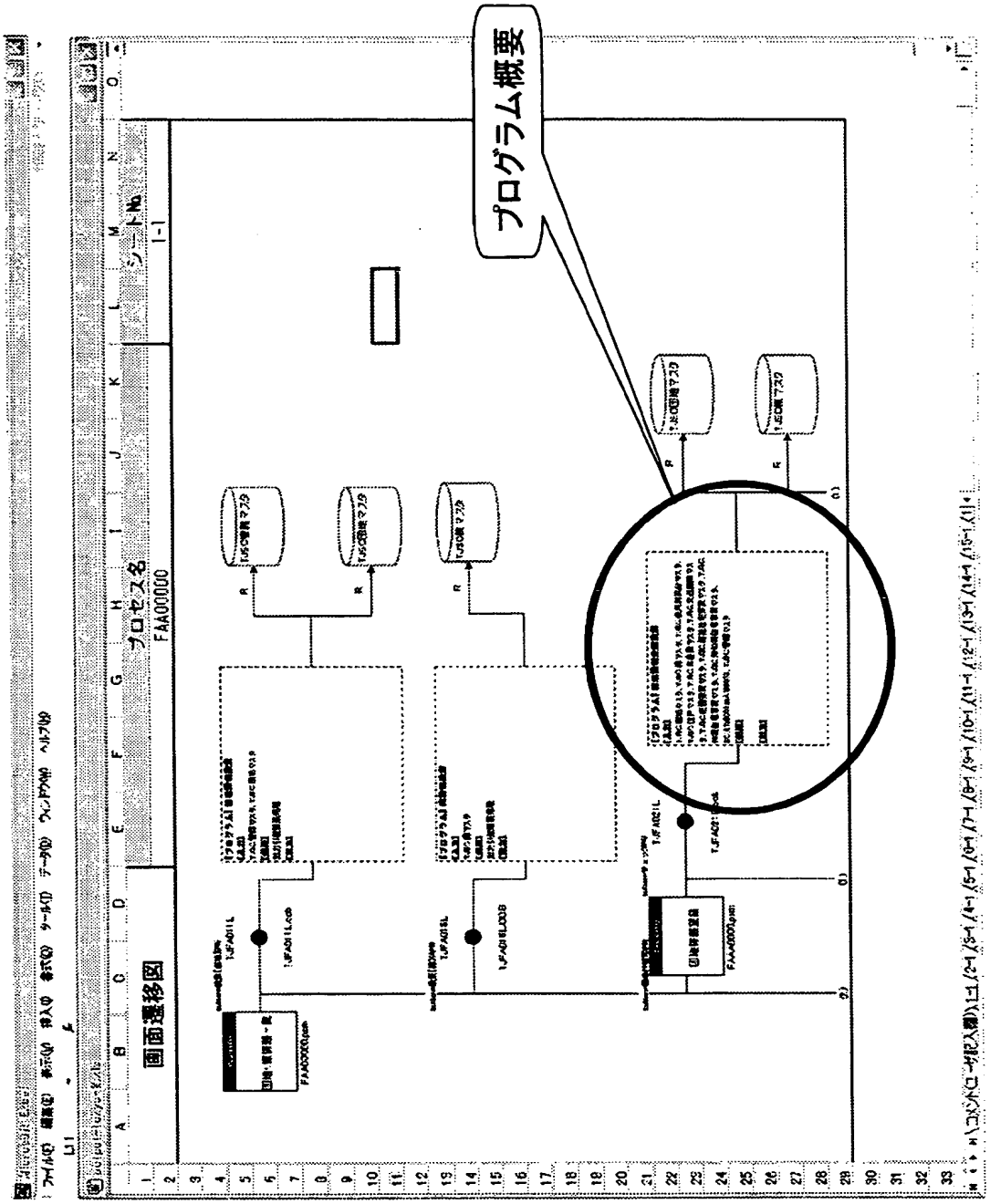
```
<Descripiton   id="00000001" name="M05000" type="display" source="M05000.psm"    layer="" layermax =""><br>
    <outline> サービス業務 </outline><br>
    <condition expression=" 選択処理=1"><br>
        <Description id="00000002" name="M05010" type="display"    source="M05010.psm"><br>
            <outline> MOD パラメータ </outline><br>
                ...<br>
        <Description/><br>
    </condition><br>
    <condition expression=" 選択処理=2"><br>
        <Description id="00000003" name="M05020" type="display"    filename="M05020.psm"><br>
            <outline> XBOST パラメータ </outline><br>
                ...<br>
        </Description><br>
    </condition><br>
</Descripiton >
```

【図５３－１】

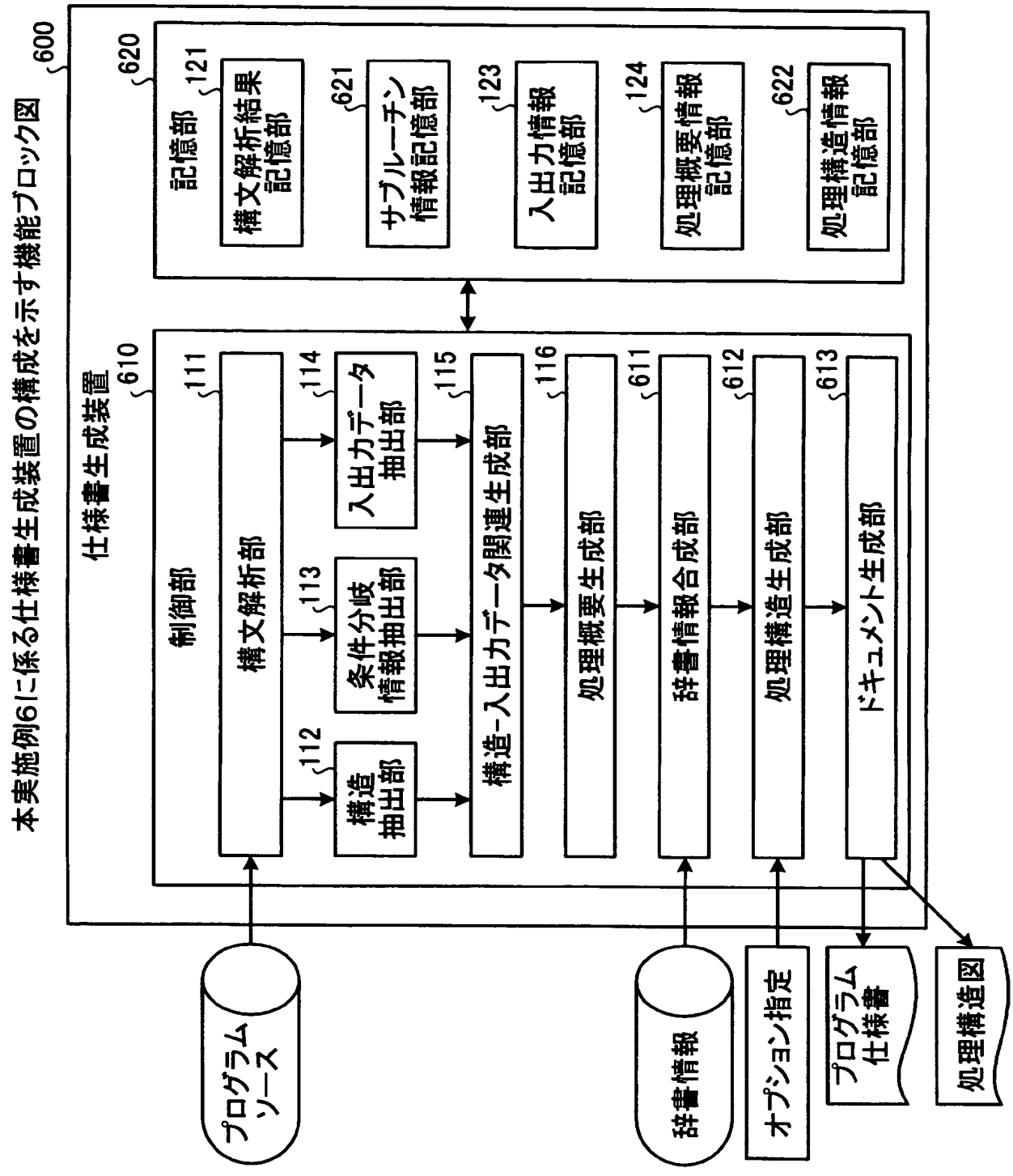

画面遷移図の一例を示す図
プロセス名
作業時間分析
No.
1-1
M05000
サービス業務
M05000.psm
選択処理＝1
M05010
MODパラメータインプット
M05010.psm
B1B08010
区分＝2
M05040
追加処理
B1B08010.cob
W
MOD パラメータ DB
区分＝2
M05050
変更処理
B1B08010.cob
RW
MOD パラメータ DB
選択処理＝2
M05020
MOD修正インプット
M05020.psm
B1B08020
B1B08020
MOD修正インプット処理(略)
B1B08020.cob
選択処理＝3
M05030
XBOST修正インプット
M05030.psm
B1B08030
区分＝1
M05060
追加処理
B1B08030.cob
W
XBOST パラメータ DB
区分＝2
M05070
変更処理
B1B08030.cob
RW
XBOST パラメータ DB

【図５３－２】

画面遷移図の他の例を示す図

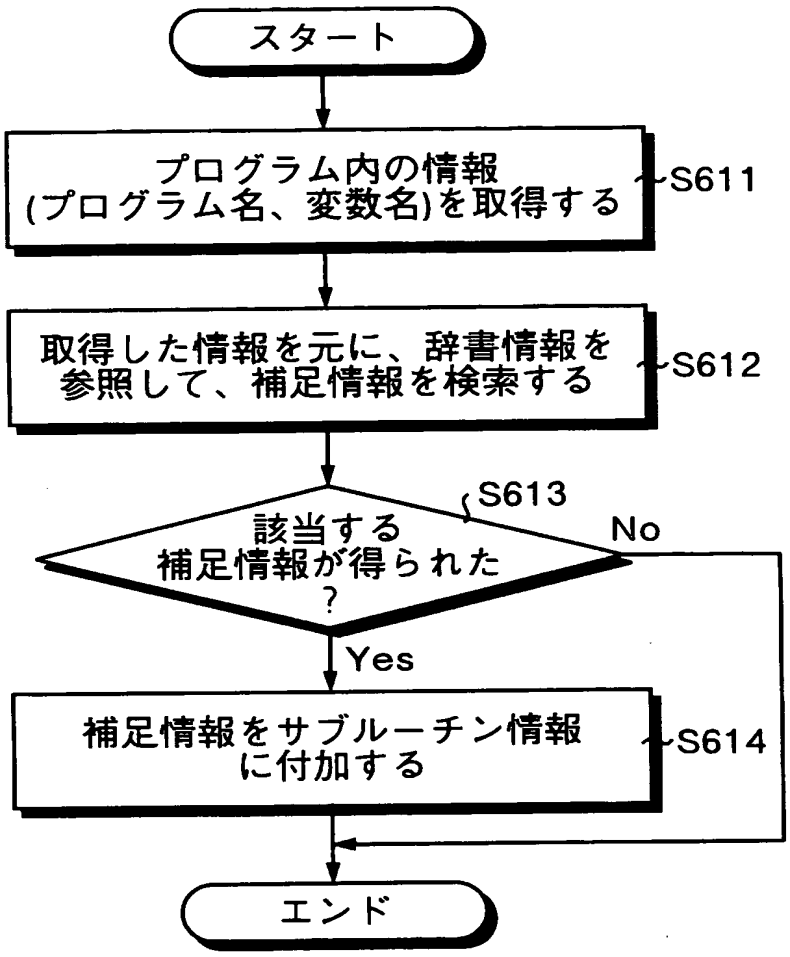


【図５４】


本実施例6に係る仕様書生成装置の構成を示す機能ブロック図
600
仕様書生成装置
610
制御部
111
構文解析部
112
構造抽出部
113
条件分岐情報抽出部
114
入出力データ抽出部
115
構造-入出力データ関連生成部
116
処理概要生成部
611
辞書情報合成部
612
処理構造生成部
613
ドキュメント生成部
620
記憶部
121
構文解析結果記憶部
621
サブルーチン情報記憶部
123
入出力情報記憶部
124
処理概要情報記憶部
622
処理構造情報記憶部
プログラムソース
辞書情報
オプション指定
プログラム仕様書
処理構造図

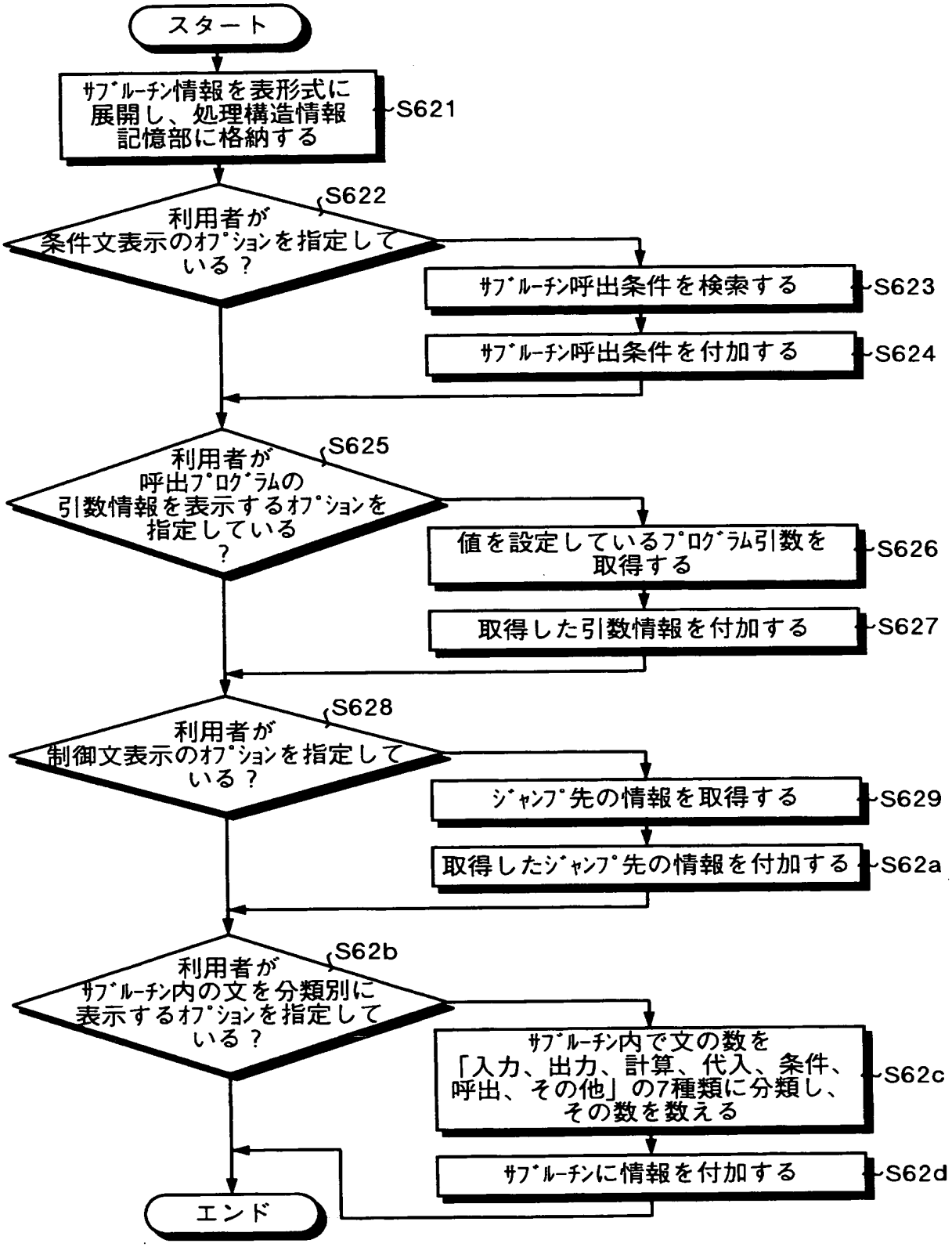
【図５５】

辞書情報合成部の処理手順を示すフローチャート



【図５６】

処理構造生成部の処理手順を示すフローチャート



【図５７－１】

処理構造図を生成する対象プログラム(呼出元)を示す図

```
PROGRAM-ID PROGA
************ PROCEDURE DIVISION ******************************
PROCEDURE DIVISION.

    OPEN    INPUT   INFILE.
    DISPLAY "## プログラムスタート " UPON CONSOLE.

    PERFORM   READ-SECT.
    IF END-FLAG  = "END"
        DISPLAY "## 一つもレコードを読めません。 " UPON CONSOLE
        DISPLAY "## エラー終了しました。 "         UPON CONSOLE
        STOP RUN
    END-IF.

    PERFORM   MAIN-SECT  UNTIL   END-FLAG = "END"

    CLOSE   INFILE.
    DISPLAY "○出力件数        =[" OUT-COUNT  "]" UPON CONSOLE.
    DISPLAY "正常終了しました。 "     UPON CONSOLE.

    STOP RUN.
************ READ ROUTINE ************************************
READ-SECT    SECTION.
    READ    INFILE
      AT END        MOVE  "END" TO   END -FLAG
      NOT AT END    ADD    1    TO   IN -COUNT
    END-READ.
READ-SECT-END.
    EXIT.
************ MAIN ROUTINE ************************************
MAIN-SECT    SECTION.
    IF FAMILYDATA = ZERO
        ADD     1       TO      SKIP -COUNT
    ELSE
        ADD     1       TO      PROC -COUNT
        PERFORM VARYING I FROM 1 BY 1
                        UNTIL I > 20 OR F -MEMBER(I) = ZERO
            MOVE    MEMBERCODE    TO  PARA1 -MEMBERCODE
            CALL     PROGW   USING PARA1
            ADD     1             TO  OUT -COUNT
        END-PERFORM
    END-IF.

    PERFORM   READ-SECT.
MAIN-SECT-END.
    EXIT.
```

【図５７－２】

処理構造図を生成する対象プログラム(呼出先)を示す図

```
PROGRAM-ID PROGW
************  PROCEDURE DIVISION ******************************
PROCEDURE DIVISION USING PARA1.

    OPEN    OUTPUT  OUTFILE.
    MOVE    PARA1         TO  OUT-RECORD.
    WRITE   OUT-RECORD
    CLOSE   OUTFILE.

    STOP RUN.


************ READ ROUTINE ***************************************
```

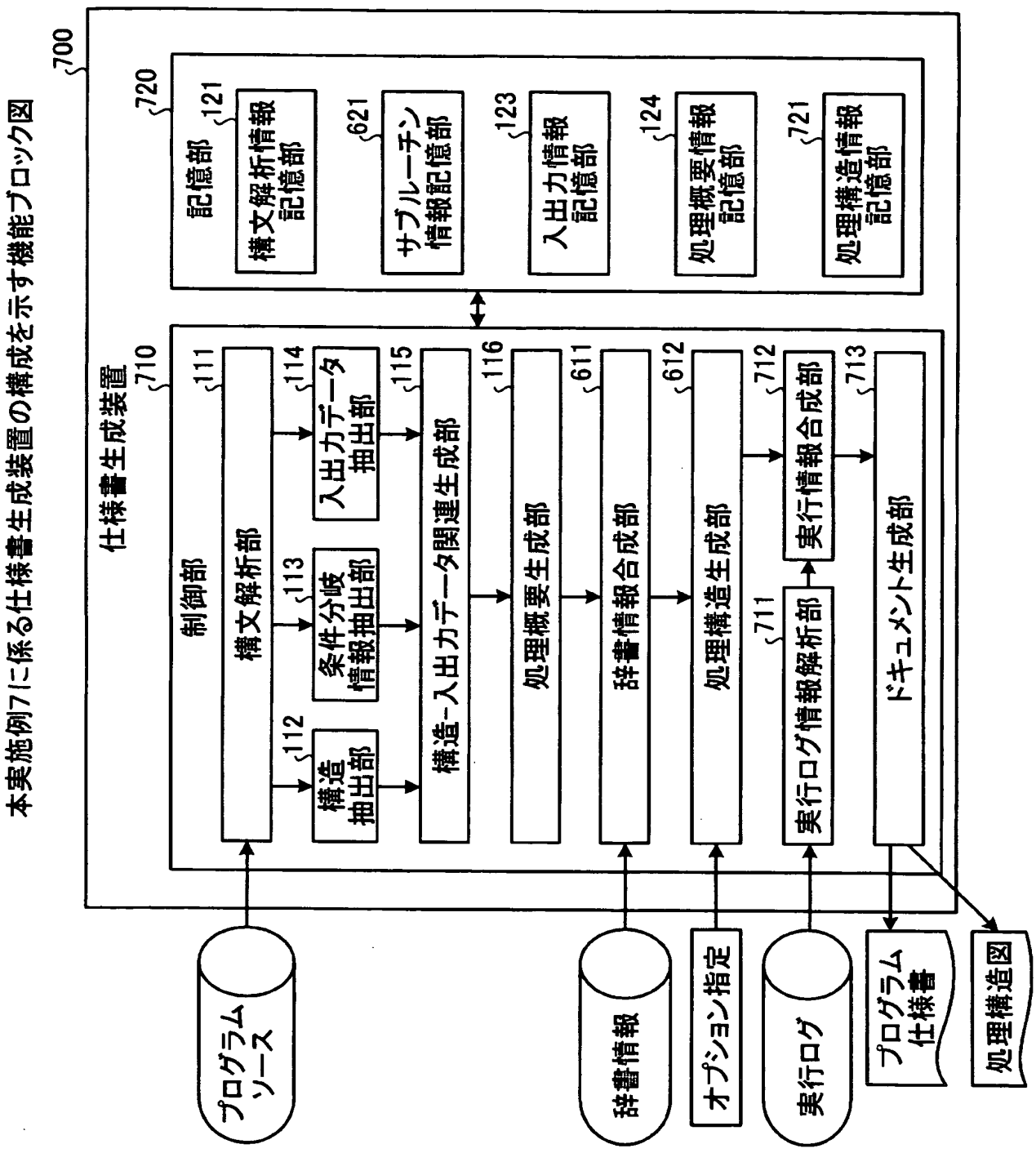
図57-1および図57-2に示したプログラムから
生成される処理構造図の一例を示す図

文書名	処理構造図

プログラム名	SECTION(1層目)	SECTION(2層目)	SECTION(3層目)	読み取りファイル	書き込みファイル
PROGA					
	first section (no name)				
		READ-SECT		INF(INFILE)	
		MAIN-SECT			
			CALL ""PROGW""		OTF(OUTFILE)

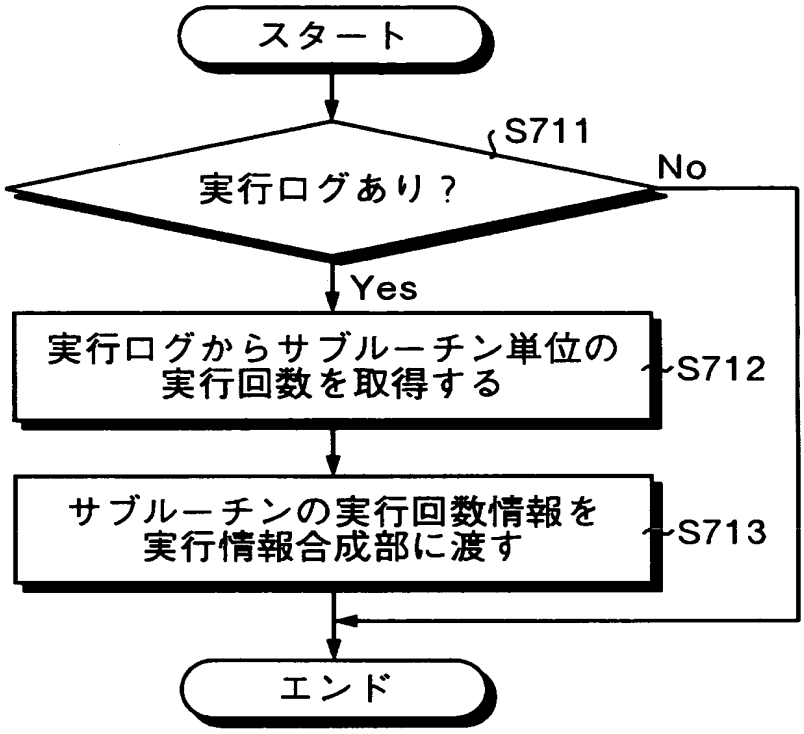
【図５９】

本実施例7に係る仕様書生成装置の構成を示す機能ブロック図



【図６０】

実行ログ情報解析部の処理手順を示すフローチャート



【図６１】

実行情報合成部の処理手順を示すフローチャート

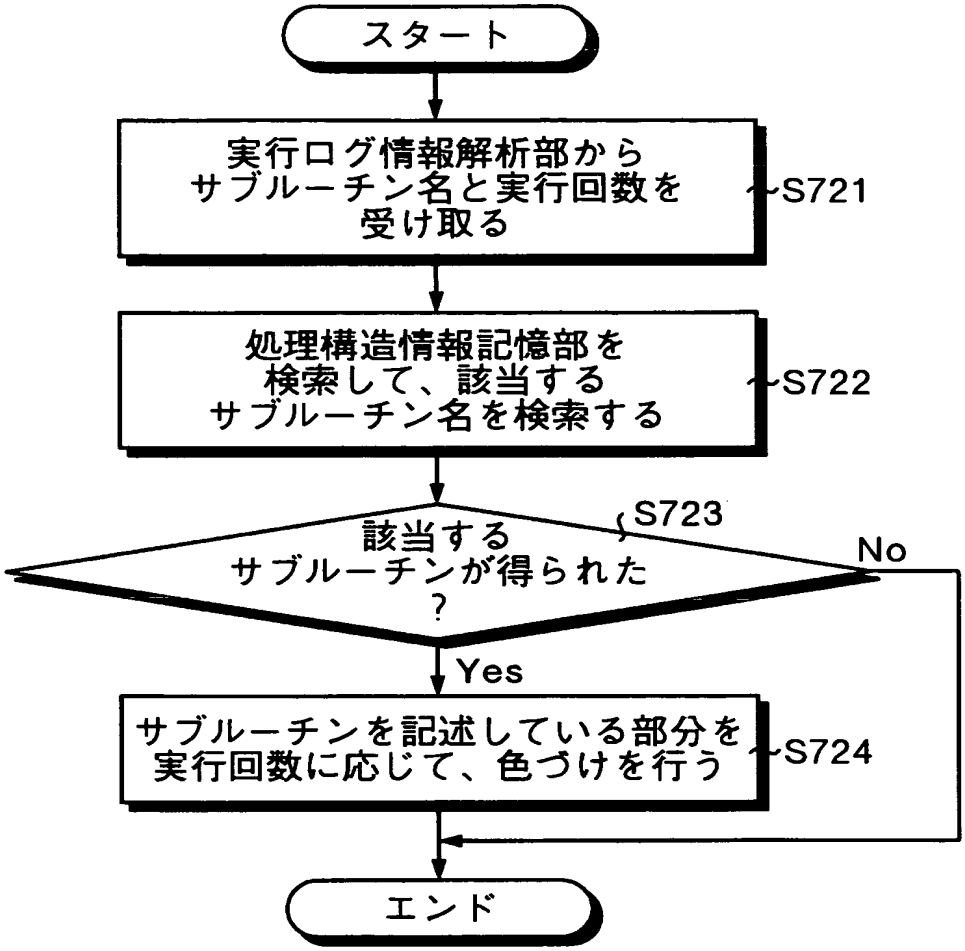


図57-1および図57-2に示したプログラムから生成される処理構造図の一例を示す図

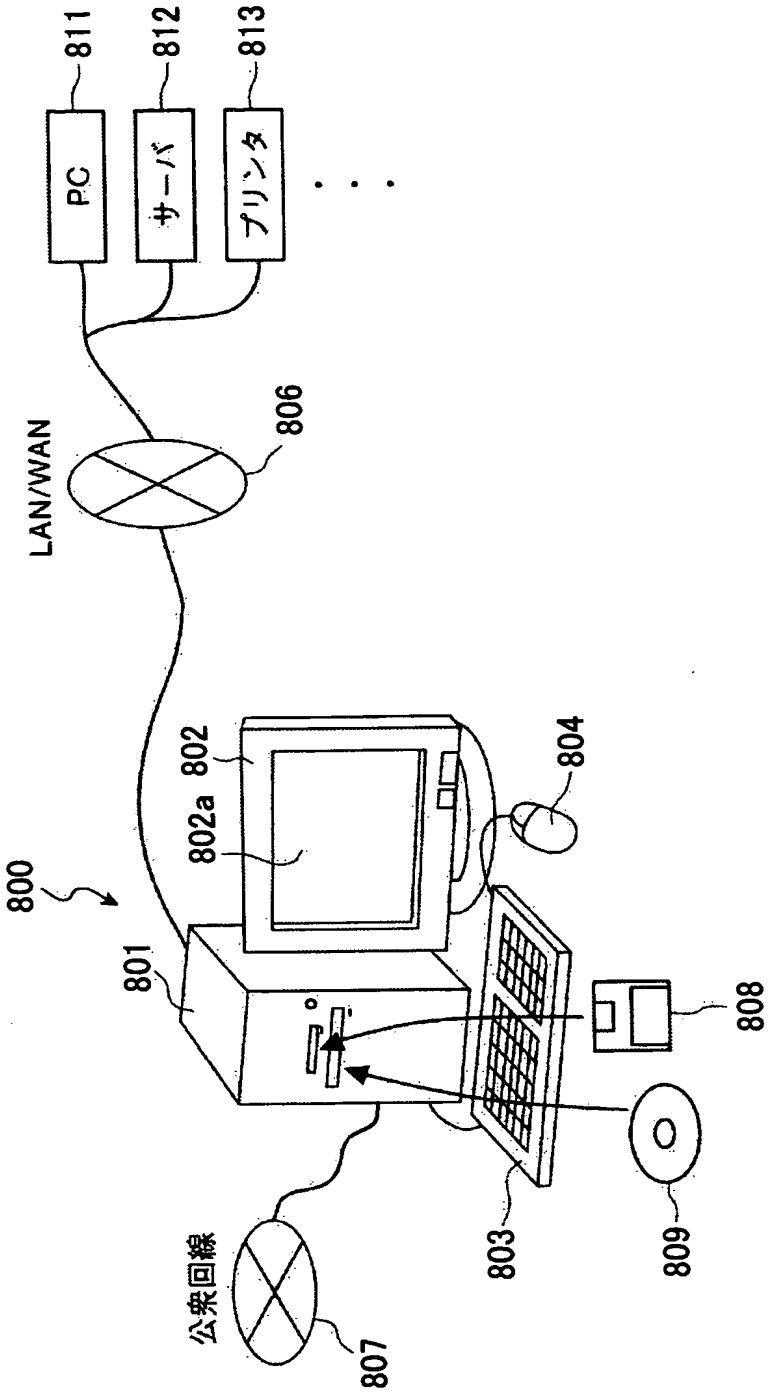
文書名	処理構造図

プログラム名	SECTION(1層目)	SECTION(2層目)	SECTION(3層目)	読み取りファイル	書き込みファイル
PROGA					
	first section (no name) 分類別ステートメント数 [入力:1,出力:5,呼出:2,条件:1, その他:9]				
		READ-SECT 分類別ステートメント数 [入力:1,計算:1,代入:1]		INF(INFILE)	
			<READ-SECT-END>		
		MAIN-SECT 実行条件 UNTIL(END-FLAG = "END")			
			CALL ""PROGW"" USING [PARA1-MEMBERCODE=MEMBERCODE]		OTF(OUTFILE)
		分類別ステートメント数 [計算:3,呼出:3,代入:1,条件:1]	<MAIN-SECT-END>		

実行情報　1回
2回

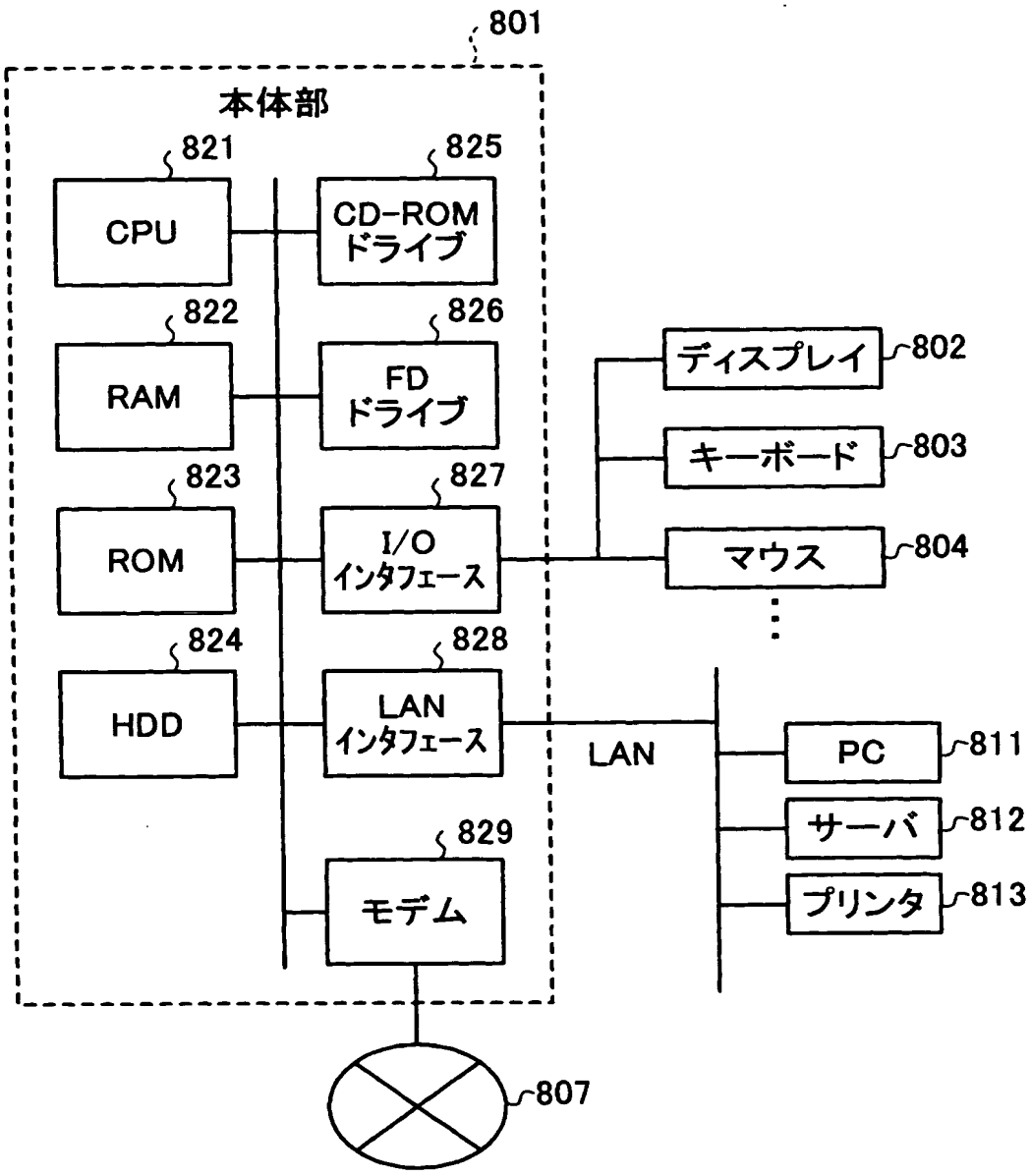
【図６３】

本実施例1～7に係る仕様書生成プログラムを実行するコンピュータシステムを示す図



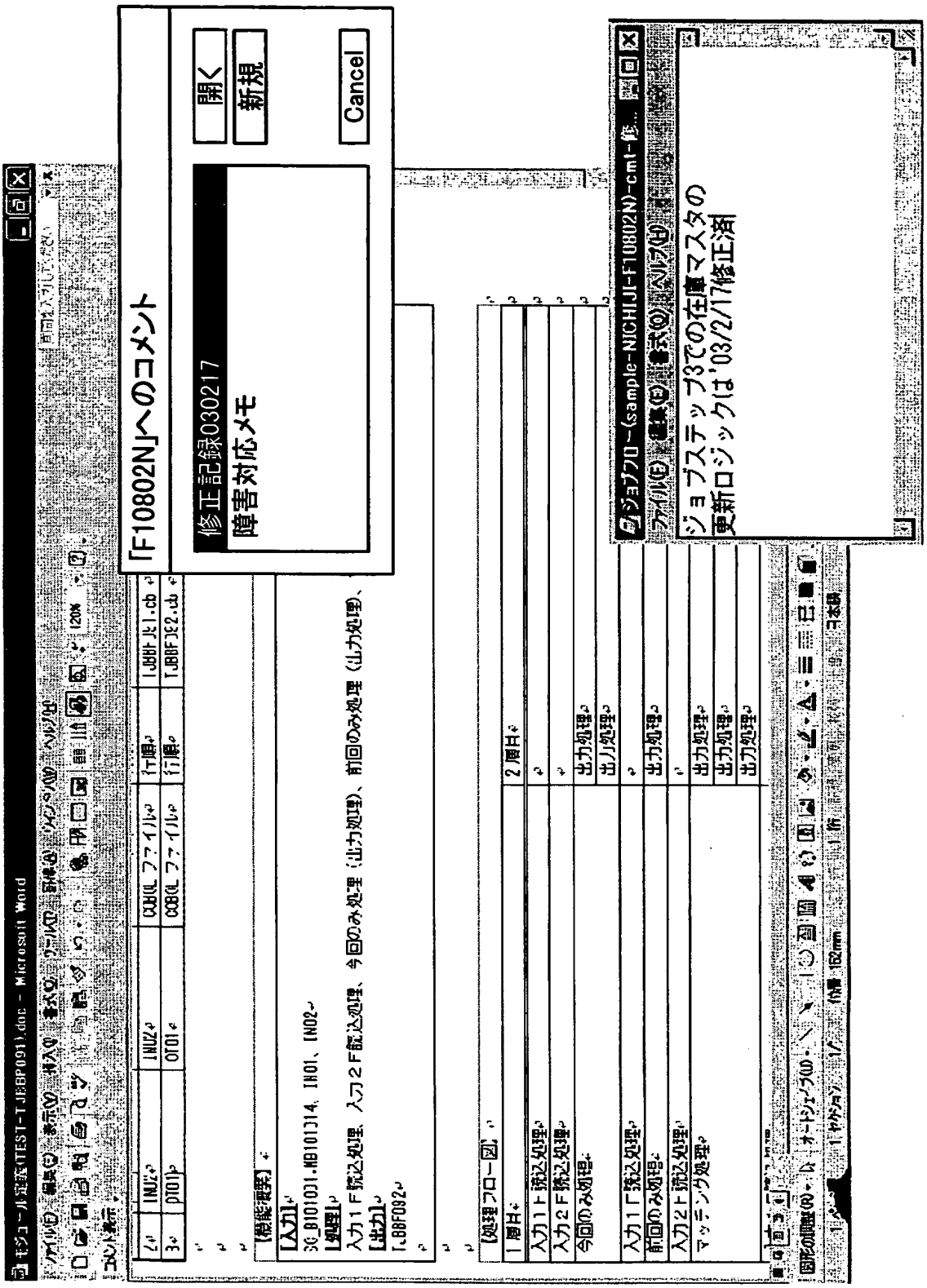
【図６４】

図64に示した本体部の構成を示す図



【図６５】

利用者記入欄を設けないコメント継承の一例を示す図



【書類名】要約書
【要約】
【課題】ソースプログラムを解析して仕様書を生成する仕様書生成装置でソフトウェアの処理の全体像の把握を容易にする仕様書を生成すること。
【解決手段】構文解析部１１１による構文解析結果から、条件分岐情報抽出部１１３が条件分岐の条件情報を抽出し、構造抽出部１１２がサブルーチンの呼出構造を条件情報とともに抽出し、入出力データ抽出部１１４が各サブルーチンの入出力情報を抽出する。そして、構文－入出力データ関連生成部１１６が条件分岐の条件情報、呼出構造情報および入出力情報に基づいてプログラムの呼出構造と入出力情報の関連づけを行い、処理概要生成部１１６が構文－入出力データ関連生成部１１６により関連づけられた情報から指定された範囲の情報を抽出して処理概要情報を作成し、ドキュメント生成部１１７が処理概要生成部１１６により生成された処理概要情報に基づいてプログラム仕様書を作成する。
【選択図】　　図１

特願２００３－３５５６９５

出　願　人　履　歴　情　報

識別番号　　［０００００５２２３］

１．変更年月日　　１９９６年　３月２６日
［変更理由］　　住所変更
住　所　　神奈川県川崎市中原区上小田中４丁目１番１号
氏　名　　富士通株式会社